# MENON
# Automating a Socratic Teaching Model for Mathematical Proofs

**Dimitra Tsovaltzi**

Saarbrücken, 2010

Dissertation zur Erlangung des Grades des Doktors der Ingenieurwissenschaften der Naturwissenschaftlich-Technischen Fakultäten der Universität des Saarlandes

| | |
|---|---|
| **Dekan** | Prof. Dr. Joachim Weickert, Universität des Saarlandes, Saarbrücken |
| **Vorsitzender** | Prof. Dr. Raimund Seidel, Universität des Saarlandes, Saarbrücken |
| **Gutachter** | Prof. Dr. Jörg Siekmann, Universität des Saarlandes, Saarbrücken |
| | Prof. Dr. Roland Brünken, Universität des Saarlandes, Saarbrücken |
| | Dr. Bruce M. McLaren, Carnegie Mellon University, Pittsburgh |
| | Dr. Colin Matheson, the University of Edinburgh, Edinburgh |
| **Beisitzer** | Priv.-Doz. Dr. Helmut Horacek |
| **Kolloquium** | 11. März 2010 |

"And it won't be as a result of any teaching that he'll have become knowledgeable: he'll just have been asked questions, and he'll recover the knowledge by himself, from within himself."

*Menon*, Plato (85d).

# Acknowledgments

I would like to thank from the heart my advisor Prof. Dr. Jörg Siekmann who gave me the opportunity to do this research and supported me throughout the years towards its completion, both intellectually and morally. From the beginning of my PhD research and until now, I have always admired his sharp scientific grasp, as well as his open mind and his readiness to endorse the interests of other people. I am indebted to the members of my PhD committee Dr. Bruce M. McLaren and Dr. Colin Matheson for their meticulous and insightful feedback on the PhD that was beyond my expectations. Both of them, each one in his own way, have been a source of inspiration and motivation to continue in the demanding arena of scientific research. I would also like to sincerely thank Prof. Dr. Roland Brünken who generously agreed to be a member of my PhD committee. I look up to his research in educational psychology.

I have had the luck to work in two research groups during my PhD research. I have felt at home and have thoroughly enjoyed working in both of them. In particular, I would like to thank the Omega group for their willingness to provide help whenever I needed it, and the ActiveMath group for giving me a future perspective that motivated me to carry the PhD through. I would especially like to express my gratitude to Dr. Armin Fiedler who believed in me, invited me to the Omega group and helped me in my very first steps as an independent researcher. I have learned a lot from him, from his intellectual clarity and scientific integrity. Special thanks also to Dr. Chad Brown and Dr. Helmut Horacek who sacrificed their precious time to review this work and provide acute and critical comments.

I cannot be grateful enough to my mother, my sisters and their families in Greece who have never seized to support me and wholeheartedly embrace my successes and failures, my happy and sad moments.

My husband, Florian, and my daughter, Sophia, have had to tolerate me, or my necessary absence in the past years. I thank them for all their patience, for accepting my aspirations and for their love that drives me at difficult times.

# Brief Abstract

This thesis presents an approach to adaptive pedagogical feedback for arbitrary domains as an alternative to resource-intensive pre-compiled feedback, which represents the state-of-the-art in intelligent tutoring systems today. A consequence of automatic adaptive feedback is that the number of tasks with pedagogical feedback that can be offered to the student increases, and with it the opportunity for practice. We focus on automating different aspects of teaching that together are primarily responsible for learning and can be integrated in a unified natural-language output. The automatic production and natural-language generation of feedback enables its personalisation both at the pedagogical and the natural-language dialogue level. We propose a method for automating the production of domain-independent adaptive feedback. The proof-of-concept implementation of the tutorial manager `Menon` is carried out for the domain of set-theory proofs.

More specifically, we define a pedagogical model that abides by schema and cognitive load theory, and by the synergistic approach to learning. We implement this model in a Socratic teaching strategy whose basic units of feedback are dialogue moves. We use empirical data from two domains to derive a taxonomy of tutorial-dialogue moves, and define the most central and sophisticated move *hint*. The formalisation of the cognitive content of hints is inspired by schema theory and is facilitated by a domain ontology.

# Kurzzusammenfassung

Die vorliegende Arbeit präsentiert eine Annäherung an adaptives pädagogisches Feedback für beliebige Domäne. Diese Herangehensweise bietet eine Alternative zu ressource-intensivem, vorübersetztem Feedback, dass das heutige "state-of-the-art" in intelligenten tutoriellen Systemen ist. Als Folge können zahlreiche Aufgaben mit pädagogischem Feedback für die Praxis angeboten werden. Der Schwerpunkt der Arbeit liegt auf der Automatisierung verschiedener Aspekte des Lehrprozesses, die in ihrer Gesamtheit wesentlich den Lernprozess beeinflussen, und in einer einheitlichen Systemausgabe Natürlicher Sprache integriert werden können. Die automatische Produktion und die Systemgenerierung von Feedback in Natürlicher Sprache ermöglichen eine Individualisierung des Feedback auf zwei Ebenen: einer pädagogischen und einer dialogischen Ebene. Dazu schlagen wir eine Methode vor, durch die adaptives Feedback automatisiert werden kann, und implementieren den tutoriellen Manager `Menon` als "proof-of-concept" beispielhaft für die Domäne von Beweisen in der Mengentheorie.

Konkret definieren wir ein pädagogisches Modell, das sich auf Schema- und Kognitionstheorie sowie auf die synergetische Herangehensweise an Lernen stützt. Dieses Modell wird in einer Sokratischen Lehrmethode implementiert, deren basale Feedback-Elemente aus Dialogakten bestehen. Zur Bestimmung einer Taxonomie Tutorielle-Dialogakte sowie des zentralen und komplexen Dialogakts *hint* (Hinweis) wenden wir empirische Daten aus zwei Domänen an. Die Formalisierung des kognitiven Inhaltes von Hinweisen folgt der Schematheorie und basiert auf einer Domänenontologie.

# Abstract


Adaptive teaching strategies that take the needs of different learners into account are a prerequisite for successful intelligent tutoring systems. They should ideally incorporate elements of all facets of teaching that together are responsible for learning. One such facet is natural language tutorial dialogue, which is crucial to the application of teaching strategies by human teachers.

Although there is an increasing amount of research on teaching strategies and on tutorial dialogues, an analysis of the multiple facets of feedback in the context of a tutorial dialogue remains to be investigated. State-of-the-art approaches do not do justice to either of these aspects. Commonly, intelligent tutoring systems concentrate on cognitive and domain-specific content and resort to pre-compiled feedback. This feedback is either derived from learning theories, or is dictated by human tutors. The feedback is then associated with pre-defined input for specific tasks, and is presented to the student as canned or template-based natural-language output in fixed sequences. As a result, adaptivity and non-cognitive aspects of feedback are sacrificed for precision of cognitive and domain-specific content designed for individual tasks. Moreover, these menu-based variants of dialogue management do not preserve the characteristics of natural-language dialogue that make it effective, such as dialogical and rhetorical structure, expressive power, and mixed initiative of the collocutors.

This thesis presents techniques to automatically integrate different aspects of teaching in a unified natural-language output. The automatic production and natural-language generation of feedback enables its personalisation both at the pedagogical and at the natural-language dialogue level. As a result, feedback can be tailored to the needs of the individual students and the discourse context. Our approach moves away from resource-intensive pre-compiled feedback and towards producing adaptive feedback for arbitrary tasks. As a consequence, the number of tasks with pedagogical feedback that can be offered to the student increases, and with it the opportunity for practice.

We propose a method for automating adaptive feedback and implement the tutorial manager `Menon` as a proof-of-concept for the domain of set theory proofs. More specifically, we define a pedagogical model that abides by schema theory and cognitive load theory, and by the synergistic approach to learning. We implement this model in a Socratic teaching strategy whose basic units of feedback are dialogue moves. We use empirical educational data from two domains: an existing corpus on electricity and electronics, and a corpus which we collected on tutoring set-theory proofs. Based on that data we derive a taxonomy of tutorial-dialogue moves, and define the most central and sophisticated move *hint*. The formalisation of the cognitive content of hints is inspired by schema theory and is facilitated by a domain ontology. The domain ontology extends the knowledge representation of the mathematical assistance system Omega for the purposes of tutorial dialogues.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 The Research Topic

The implementation of teaching strategies within Intelligent Tutoring Systems (ITSs) remained until recently underrepresented in the community. A lot of theoretical work on teaching strategies exists and their necessity is well recognised. Moreover, human tutors, who make extensive use of teaching strategies, are considered the model for ITSs especially since an influential paper [Bloom, 1984] that revealed the effectiveness of one-to-one human tutoring as opposed to teaching in the classroom. However, the complexity of teaching strategies makes them a less successful branch of research in ITSs even to date, especially with regard to their automation.

This thesis concentrates on automating feedback as a means for implementing adaptive teaching strategies. It explores the possibility of integrating automatic feedback in a natural language (NL) dialogue tutorial system. Empirical evidence has shown that NL dialogue capabilities are a crucial factor to making human explanations effective [Moore, 1993; Core *et al.*, 2000; Porayska-Pomsta and Pain, 2000; DiPaolo *et al.*, 2004; Carenini and Moore, 2006]. Such capabilities include, for instance, the possibility to refer to previous dialogue chunks, to use appropriate discourse[1] markers, to include basic dialogue phenomena such as acknowledgements and back-channel feedback in general, and to employ multiturn tutoring feedback. In the genre of tutorial dialogue alone back-channel feedback that is sensitive to the student's input is provided at a rate of 2.71 per minute by human tutors, as was observed in a corpus of human-to-human tutorial dialogues [Rajan *et al.*, 2001].

Moreover, the use of teaching strategies that abide by active learning is becoming a prerequisite for tutoring systems. Such strategies are often referred to as dialectic or *Socratic* [Stevens and Collins, 1977] and are contrasted with the more traditional didactic tutoring style characterised by long explanations,

[1] We use the term "discourse" to refer to goal-targeted conversation and the term "dialogue" to refer to free conversation.

where the tutor has control of the tutoring and the learner is regarded as the passive recipient of knowledge [Rosé *et al.*, 2001b]. Socratic strategies, on the contrary, are student-oriented. They use a dialectic form in order to instruct the student and monitor the learning process, and they put emphasis on eliciting as much information from the student as possible. Socratic strategies have been demonstrated several times to be superior to pure explanations, especially regarding their long-term effects and transfer of knowledge [Chi *et al.*, 1994; Rosé *et al.*, 2001b; Ashley *et al.*, 2002; DiPaolo *et al.*, 2004]. Researchers hypothesise that requesting students to use NL input forces them to self-explain what they are doing, which makes them think harder. Consequently, an increasing, though still limited number of state-of-the-art tutoring systems investigate natural-language interaction and automatic teaching strategies, normally including some notion of hints, which are thought to implement the Socratic style of teaching. Additionally, problem solving is known to promote learning [Delclos and Harrington, 1991; Wilson and Cole, 1991; Lim *et al.*, 1996; Wu, 2001], and high student control during problem solving of the kind that hints can foster has a motivational effect [Keller, 1987; Weiner, 1992; de Vicente and Pain, 1998]. Finally, hints can prevent students from taking wrong paths while learning actively and direct them to productive paths [DiPaolo *et al.*, 2004].

Although some work on hinting exists [Hume *et al.*, 1996b; DiPaolo *et al.*, 2004; Murray *et al.*, 2004; Horacek, 2006], there is on the whole a lack of research in the area of automatically adaptive feedback. The standard pre-compiled feedback does not do justice to the role of hints in learning. Hints should ideally take the student's frame of mind into account in order to address the relevant piece of information at each point and trigger active learning. They should also be tailored to the need of the student for affective support. Incorporating this kind of adaptivity into an ITS faces, of course, the problem of reasoning about the student's frame of mind and affective state. However, it becomes impossible when using pre-compiled feedback. Moreover, in providing canned NL feedback, the adaptivity of human tutoring that is owed to expressing oneself in natural language is lost. With it natural argumentation also suffers, which is characterised by human-like reasoning, dialogical and rhethorical structure, expressive power, and effectiveness of argument [Reed and Grasso, 2004]. A big part of adaptivity in tutorial dialogue for anything but the most simple domains and tasks is also the possibility that the student can take the initiative in solving the task, for example by asking a question, and the tutor leads the student through multistep feedback tailored to the question [Freedman, 2000]. Therefore, there is a need for dialogue-based ITSs that allow mixed initiative and provide automatic feedback for free student input that adapts depending on the tutoring and dialogue context.

The high value of providing a greater variety of feedback strategies in a flexible NL dialogue context, as opposed to using fixed sequences, and the significance of the thorough investigation of hinting are well recognised in the field [du Boulay and Luckin, 2001; VanLehn, 2006]. The fact that there is room for research in integrating dialogue theories and educational teaching strategies into tutorial dialogue approaches for ITSs is also common knowledge in the community [Zinn, 2002]. More specifically, further research is needed to understand the multifunctionality of hints, the way these multiple functions are combined in one hint, the connection of hints to other forms of pedagogical feedback and to dialogue moves (DMs), and their place in tutorial dialogues.

This thesis investigates the possibility of automating a Socratic teaching strategy. It defines a teaching model based on prominent learning theories. It investigates pedagogical feedback for realising the model (e.g. motivational feedback) and scrutinises hints as the main and most demanding characteristic of this model. *Hints* are defined as the means for encouraging active learning. They take the form of eliciting information that students are unable to access without the aid of prompts, or information that students can access, but are unaware of its relevance for the problem at hand. A hint can also point to an inference that students are expected to make based on knowledge available to them [Hume *et al.*, 1996b; DiPaolo *et al.*, 2004]. The thesis undertakes the research of hints in the context of problem solving and NL tutorial dialogue. It presents a way of automatically producing student, situation, and discourse sensitive hints.

In terms of pedagogical consideration that the Socratic teaching strategy takes into account, we use four main theoretical models for defining the tutorial model that the strategy implements. First, we subscribe to schema theory and the definition of schemata as heuristic models, which can be re-applied and modified for solving more or less similar problems respectively. They are learned via experience in the domain [Delclos and Harrington, 1991]. Second, we take into account considerations deriving from cognitive load theory, which is based on research on working memory capacity and influences schema acquisition [Owen and Sweller, 1985]. We use this theory to guide the way we regulate the load imposed on the student in order for learning to occur. Third, we model aspects of motivation theory concerned with how much effort students put into learning [Keller, 1987]. Fourth, we follow a synergistic approach to learning by promoting primarily implicit and some explicit learning, when necessary [Sun *et al.*, 2001].

An evaluation of the teaching strategy and the automatically produced feedback that we implement gave positive results. More specifically, 4 out of 5 mathematicians who evaluated our teaching strategy and were either teachers or had strong pedagogical background preferred `Menon`'s feedback to equivalent feedback of a strategy that was previously shown to effect learning.

In this chapter, we first give a general overview of ITSs and an account of the current state in our general research area in Section 1.2. We then define our research problem in Section 1.3, our specific goals and scientific contributions in Section 1.3.6. We present our methodology for achieving these goals in Section 1.4 and an example that illustrates our overall approach 1.3.4.1.

## 1.2 General Overview and State of the Art

ITSs are systems that use computer support and AI techniques to simulate or assist teaching. Developers of ITSs aim to have their systems participate in teaching and contribute to various facets of it, either by offering technological support, or by increasing the tutoring hours per student. Assistance can be provided to a student, a group of students, or a teacher.

The following three main research techniques have been commonly applied in building ITSs, which have helped derive guidelines for what the behaviour of the ITS should be [du Boulay and Luckin, 2001]:

1. Observing and simulating the behaviour of human tutors [Graesser *et al.*, 2001]

2. Reviewing learning theories and deriving a teaching theory [Anderson *et al.*, 1995]

3. Observing students and deriving a teaching theory [VanLehn *et al.*, 2005]

There are various aspects of learning and teaching that are simulated or dealt with by ITSs. The following areas of research constitute development choices for simulating the required behaviour that have become standard throughout the years. AI techniques are used to implement this behaviour and to meet the challenges that each of the above areas of ITS development presents.

**Learning Environment** This is the environment in which the user works. It is the *sine qua non* of ITSs. It provides an interface for solving the task and for the interaction between student and tutor. It also provides computer tools for helping the student solve the task and facilitate the interaction. The student tools included in a learning environment may comprise graphical support for plotting and organising knowledge, natural language input, representation of the learner's model (often referred to as "open learner model" [Dimitrova *et al.*, 1999]), automatic calculations of values, synchronised actions (collaborative systems) etc. [Koedinger and Anderson, 1997; VanLehn *et al.*, 2005]. For the teacher, typically authoring tools are provided, which allow teachers to author the teaching material of the ITS to fit the needs of their class, or the goals of their preferred teaching model [Murray, 2003]. Another support for the teacher is monitoring tools, which may provide an overview of the student's or the class's activities and hence support curriculum planning, and the choice of feedback [Groot *et al.*, 2007]. The interface may also include natural language dialogue capabilities for accepting NL input from the student and formulating the teacher's output [Evens *et al.*, 2001; Jordan *et al.*, 2006].

**Knowledge Representation** The knowledge representation in ITSs concerns the domain knowledge that the ITS teaches. This must be represented in a way that makes it possible to assess the knowledge of the student, to decide which knowledge must be taught further, and to present this knowledge to the student.

The first of these issues relates to student modelling. The other two relate to teaching strategies. Moreover, knowledge has to be represented efficiently, so that a large number of tasks can be handled by the ITS given the same knowledge representation and small implementation effort. An example of this are systems that can generate their own tasks from the knowledge representation (e.g [Mitrovic, 2003; Ullrich *et al.*, 2006]). Knowledge representation also has to cater for the needs of the learning theory and the teaching strategy that the ITS uses. To this aim, developers of ITSs employ teachers who are responsible for representing both the tasks and the best feedback for potential errors while solving these tasks (e.g. [VanLehn *et al.*, 2005]). Authoring tools mediate this procedure by accepting user friendly input and translating it into the representation that the ITS uses.

The granularity with which knowledge is represented is commonly one of the pitfalls of knowledge representation [VanLehn, 2006]. If too big sub-steps for solving a task are assumed, errors in finer steps will be overlooked. If too small steps are represented, abstracting away from detail is difficult. Such abstraction is often necessary to avoid overloading students and allow them to concentrate on the specific objectives of the lesson. The granularity of the represented knowledge also restricts the feedback, which has to be of the same granularity. In a physics task, for instance, if the arithmetic involved is not represented, but only the reasoning in terms of laws of physics, then errors caused at the math level will not be tracked down and cannot be remediated. This however might be the objective of an ITS that teaches the laws of physics and wants to spare students the details of the math.

A knowledge representation that is the aggregate of a representation for every task is not efficient, but it allows providing feedback tailored to every problem based on the teaching strategy of choice. An efficient knowledge representation abstracts from the specific tasks to the effect that some of the uniqueness and hence the quality of the feedback may be lost. These two concerns pose a great challenge to ITSs: To represent knowledge efficiently and at the same time in a way that it can enable pedagogically appropriate feedback for every task. Knowledge must be represented so that it can be used as the domain content of feedback no matter what teaching strategy this feedback implements. An example of such an approach is the ActiveMath system which uses the OmDoc (Open Mathematical Documents) markup language to represent domain knowledge as a uniform interface for structured theories [Kohlhase, 2006].

**Student Modelling** Student modelling is a specific subarea of user modelling which concerns itself with formalising attributes that play a role in learning so that they are readily available for assisting teaching [Corbett *et al.*, 1997]. Such attributes may represent for each student one or more of the following general categories: the degree of mastery of knowledge, the evaluation of the student's metacognitive abilities, the estimation of the student's affective state and motivation. In general, student modelling is an attempt to operationalise those attributes. The degree of mastery of knowledge is by far the most common attribute modelled in ITSs. In operationalising them, the ITS developers define the attributes for the specific ITS. For example, degree of mastery may be defined based on how many tasks a student has solved, of what difficulty, and in how much time. These are domain-independent attributes. Another operationalisation that can be common across domains may be modelling motivation by representing for example attention, relevance, confidence, and satisfaction [Keller, 1987].

On the other hand, the operationalisation may differ a lot between domains, for instance if one wants to directly operationalise the mastery of chunks of knowledge one has to specify the relevant chunks and what it means to master them. For example, for physics tasks this may include which laws of physics a student has to know, as well as what it means to know them. It may mean to be able to write down a formula for them, or to solve tasks by applying them. Potential errors that the student may make and which the ITS developer wishes to track and remediate must also be represented. Capturing such errors is often used to operationalise of the degree of mastery. In that case, a student who commits such errors will not have mastered the related chunks of knowledge.

Due to decisions like the ones we mentioned, the operationalisation of the attributes involved in student modelling is closely intertwined with the teaching strategies and the general pedagogical approach that is implemented by the ITS, as well as with the overall knowledge representation.

**Teaching Strategies** Teaching strategies concern themselves with helping students to acquire knowledge by instruction. This presupposes choosing a domain like mathematics, physics, etc, specifying a curriculum for this domain with the specific domain knowledge that the student must acquire, and defining what it means that the student has acquired this knowledge. Teaching strategies are closely related to cognitive theories of knowledge representation and learning theories, which play an increasingly important role in defining them. General principles of learning can be outlined from such theories, which help in turn to derive general guidelines for teaching. General guidelines have limitations in that they do not cater for individual learning styles that largely affect learning process. Therefore, a main issue concerning teaching strategies is adapting the various aspects of teaching to the needs of the student.

Aspects of learning that teaching strategies aim to influence include:

1. Cognitive processes, which pertain to learning directly and to how students acquire knowledge

2. Metacognitive process, through which students can control their own learning

3. Affect and motivation, which relate to if and how much students are willing to learn. Social processes are considered here, too as the macro-level counterpart of affect and motivation

Cognitive processes have been in the focus of research on teaching strategies, while metacognitive and affective processes have only recently started getting due attention.

There are general issues that a teaching strategy must take a stance on [Corbett *et al.*, 1997; du Boulay and Luckin, 2001; VanLehn, 2006]. If and how much scaffolding one will provide is one issue. Too little scaffolding like free exploration can be demotivating for the learner, who is required to navigate through a domain or a task with no or minimum help, even when students feel they need it. Too much scaffolding, like giving hints for all steps before students have made an attempt on their own, can lead to little learning, as there might be little room for the student to make the cognitive effort that is necessary for learning to occur.

What kind of scaffolding one will provide is another issue. For example, whether scaffolding will consist of tutor feedback, or adaptation of the task difficulty. The tutor feedback can be given on request or when the tutor judges it necessary. In either case, the tutor may decide to provide help on a spectrum that includes giving simple evaluation of the student attempt (correct or incorrect), to giving a hint on improving the attempt, to giving away the answer. Which hint to give is itself a very complicated decision and involves among other issues choosing the right moment to give a hint, the right level of hint to promote learning, the domain knowledge to address, and the way to address it. This latter point often requires that the system recognises the step that the student is attempting and identifies the best next step to address. Such requirements pertain to AI planning that is an area of research in its own right.

The choice of task is based on the student model, which represents the knowledge that the student should acquire and reasons about the mastery of the domain knowledge by the student. The kind and level of task to be assigned next is instructed by the teaching theory. It involves decisions like progressing slowly through the knowledge taught or presenting a more difficult problem that challenges the student, and if the next task should deal with the same sub-part of the domain or move to a new one. A theoretical work in this direction that is favoured among ITS developers is Vygotsky's "zone of proximal development" [Vygotsky, 1978].

How one will deal with errors is a third issue concerning teaching strategies. More particularly, decisions involve if all errors will be treated, which would put emphasis on reacting to erroneous knowledge rather than constructing new knowledge. When the errors to be treated are selected, an ITS has to decide if they will be treated all together by a unified feedback or separately by giving specific feedback for each error. Moreover, one has the choice of treating errors as they occur, with the argument of preventing further confusion, or at a more opportune time through delayed feedback, to avoid distracting the student too much.

As far as metacognition is concerned, a system can promote reflection, in the sense of having students think about the task and their own actions and answers more carefully. Further metacognitive processes that can be promoted are related to self-regulated learning, where students learn to judge what they know and understand, when they need to practice more, whether they need help and what kind of help [Gama, 2004; Roll *et al.*, 2007].

Affect and motivation are also attracting more and more attention in the ITS community [Barrow *et al.*, 2008; Mavrikis *et al.*, 2003; Beal and Lee, 2005]. Issues to be considered here are, for instance, how to manage a balance between challenging students without demotivating them, how to recognise that affective support is needed, how to cater for differences in sex or culture that seem to play an important role.

**Tutorial Dialogue** One-to-one tutoring most commonly takes the form of a dialogue between the tutor and the student, where the teacher uses dialogical methods to teach the student. Such dialogical methods have a bearing on different aspects of ITSs, but they are particularly related to teaching strategies and the learning environment. Tutorial dialogues, and especially NL tutorial dialogues, are used to implement tutoring strategies, especially in an attempt to simulate the behaviour of human tutors, which typically use NL dialogue in teaching [Graesser *et al.*, 2001].

A main issue in the application of NL dialogue in the context of ITSs is handling the student input. This issue has three facets. One is the problem of NL understanding, that is, making sense of the input provided by the student, which constitutes a separate area of research. A second facet is that free student input requires sophisticated dialogue management in a difficult genre like tutorial dialogue. Finally, evaluating free student input with regard to its domain content and reasoning about appropriate feedback constitute a major bottleneck for implementing tutorial dialogues in ITSs. An efficient although simple solution to these issues has been to use menu-based dialogue. Students are required to select their input from a menu. The entries in the menu are associated with pedagogical feedback that is normally pre-formulated in NL.

Speech is also starting to be part of ITSs and especially speech production is used by pedagogical agents to demonstrate emotion by use of intonation [Graesser *et al.*, 2001].

### 1.2.1 ITSs with NL Dialogue Management and Tutoring Strategies

In this section, we describe the ITSs that have common characteristics with our research topic and concentrate on the aspects of them that are most relevant to our approach. These include pedagogical knowledge, various kinds of feedback strategies, and NL dialogue capabilities. Throughout our accounts of the systems, we point out the problematic areas that we propose to address. We cluster the descriptions based on the aspects that are more developed in the reviewed systems. We also look at theoretical work on the design of feedback strategies for interactive learning tasks, and at research on task-oriented dialogue.

#### 1.2.1.1 AutoTutor

AutoTutor is an ITS that employs conversation to teach students Newtonian qualitative physics, computer literacy and research methods [Person *et al.*, 2000; Graesser *et al.*, 2001; 2003; Person and Graesser, 2003]. It uses a talking head as a pedagogical agent with facial expressions and variations in intonation to express discourse cues. These are used as non-verbal feedback and aim at influencing the student's emotional state like human tutors do. AutoTutor also simulates the discourse strategies applied by non-expert tutors on human-to-human tutorials. Curriculum scripts are used to represent the lesson content in form of topics, didactic descriptions, tutor questions, examples, figures and diagrams.

The student's input, in the form of a short essay, is analysed with Latent Semantic Analysis (LSA), which is a probabilistic method used to assess the student answer. The pedagogical feedback to this student input aims at self-explanation, which is defined as having the student articulate expected answers. Emphasis is put on active learning, in the sense of constructivism that encourages learner's to discover knowledge for themselves, and on deep reasoning. Feedback is associated with each curriculum script and the focal question that it deals with. A set of good answer aspects are defined for each focal question, together with associated feedback for eliciting or presenting these aspects. A set of anticipated bad answers for each aspect and a matching correction are also represented.

The kind of feedback used includes: (i) asking a question, e.g. to repair an error, to redirect the student's activity, to request a clarification, (ii) hints, (iii) additional examples, e.g. an easier one, (iv) reminding the student of a similar example, (v) answering a student question, (vi) rearticulating, (vii) affective feedback, e.g. commenting on the student's ability, etc. AutoTutor additionally provides back-channel feedback, that is feedback that facilitates the conversation but carries no further meaning. Moreover, positive feedback is expressed by "Yeah", "OK", "Right", and is meant to motivate the student, neutral feedback is expressed by "Uh-huh", and negative feedback by "No". Finally, the tutor gives summaries of the interaction.

Dialogue management in AutoTutor [Graesser *et al.*, 2004] handles the dialogue between the agent and the student and is specialised for the domain taught. It is based on an augmented finite state transition network, where nodes refer to knowledge goal states, normally the aspect that the student should self-explain next, or dialogue states, e.g. that the student has provided an assertion in an attempt to answer a previous question by the tutor. Arcs in the network refer to tutor dialogue moves or discourse markers complementing the tutor output. Transitions require the fulfilment of conditions, which implement the pedagogical and dialogue model. These conditions take into account the knowledge goal state, the curriculum scripts, the dialogue state, the LSA measures etc.

Dialogue management consists of four components: (i) a dialog Advancer Network, (ii) a set of fuzzy production rules, (iii) the selection of the next good answer aspect to cover, (iv) the articulation of the selected good answer aspect. The Dialog Advancer Network chooses discourse markers, tutor turn categories, and standard expressions for the realisation of the tutor turn. The second component of AutoTutor defines a set of fuzzy production rules, which incorporate knowledge about the dialogue history, and an evaluation of the student's input. The evaluation assigns a value of correctness on a correct-incorrect continuum, and assesses the coverage of good answer aspects in the student input. The fuzzy rules are a kind of feedback selection mechanism. The third component is responsible for the selection of the next good answer aspect to cover, which forms the next dialogue topic. This is based on a classification from 0 to 1 of the coverage of each good answer aspect, which is done during the LSA analysis of the student input. The next aspect to address is chosen based on pedagogical principles, like Vygotsky's zone of proximal development [Vygotsky, 1978], which chooses the aspect with the highest coverage but below a given correctness threshold. The fourth component deals with self-explanation that is realised by the articulation of the next good answer aspect by the student. AutoTutor aims to get the student to articulate each noun phrase, propositional phrase or clause in every good answer aspect. To that end, aspects are associated with a series of three hints, which are produced in an order from less to more specific until the aspect is covered.

The dialogue engine of AutoTutor follows the form-filling approach, where a form defines the information a system needs to acquire, before a topic is considered closed and allows a flexibility in the order this information is entered by the user [Person *et al.*, 2000]. Such an approach is designed for systems that mainly require strictly information from the user, like dates, or names of companies in the travel domain. The output of such systems consists of canned phrases that ask for this information and is not appropriate for producing flexible system output.

AutoTutor was evaluated in a course on computer literacy at the University of Memphis with approximately 200 students in a within-subject study. All students were subjected to three conditions for different curriculum topics. One topic was taught using AutoTutor, a second topic was taught by re-reading a chapter, and all of the remaining topics were the control and they were neither re-read nor tutored but the students had read them before. Tests for the different topics were solicited. The results showed a significant raise in learning gains for the topic that was taught by AutoTutor. Additional evaluations of the naturalness of AutoTutor's dialogue moves and quality of pedagogical feedback by experts showed that the experts found the conversation smooth as opposed to awkward and its pedagogical quality good rather than bad. In a later study, student's were asked to distinguish between dialogue moves produced by AutoTutor and real tutors. The results showed that they were not able to discriminate between the two.

One of the versions of the AutoTutor system [Graesser *et al.*, 2004] has implemented two explicit functions of hints: redirecting the student back to the topic, directing the student to a specific aspect of the expected answer. These functions are realised by either presenting facts, or asking leading questions, or re-framing the problem. This implementation exhibits an awareness that hints in the context of tutorial dialogues function on more than one dimension. However, it is restricted in two ways: It does not separate between dialogue and cognitive functions, and it does not propose a way of composing the various aspects into one hint. It thus fails to capture all functions of hints and it does not provide a structured analysis of hints for formalising them.

To evaluate this implementation, simulated student contributions were used. The evaluation showed that from a pedagogical perspective the hints were rather poor. A second evaluation was done that collected think-alouds of students working with AutoTutor. The feedback was on the whole moderate and the developers report three main challenges: (i) producing more appropriate discourse markers, (ii) disambiguating between hints to which the student is supposed to respond actively and assertions by the tutor, and (iii) improving the intonation.

The pre-compiled feedback and the dialogue management that mixes domain and dialogue considerations are restrictions on the flexibility of the feedback produced by AutoTutor on both of these aspects.

#### 1.2.1.2 CIRCSIM-Tutor

CIRCSIM-Tutor [Hume *et al.*, 1996a; Yujian *et al.*, 1999; Zhou *et al.*, 1999; Evens *et al.*, 2001; Lee *et al.*, 2002] is an intelligent tutoring system for blood circulation. Tasks set by the tutor typically provide a description of a physiological condition and require that students name the variables related to this condition and that they predict their values. CIRCSIM incorporates a lesson planner that generates topics and from them sets subtopics, which then serve as discourse tasks. The lesson planner also decides on the sequence topics and subtopics will be taught. A domain-knowledge concept map is used, which explicitly represents variables in the domain, relations, and causal links among them. An algorithm iterates through the concept map and specifies the parts of the required answer for the task at hand. The algorithm makes use of three meta-concepts, which are defined in the concept map, namely, the related parameters that have to hold for a physiological condition to be identified, same characteristics between those parameters, and distinguishing characteristics of each parameter. The parts of the answer are then elicited from the student, who learns the static domain representation this way.

A discourse planner controls the interaction between the student and the tutor. It is responsible for covering all discourse tasks that deal with the subtopics set by the lesson planner, for choosing a plan to tutor them, and for producing feedback tactics. The system implements four types of plans, each with a corresponding feedback tactic:

1. *tutor*, associated with asking questions
2. *give answer*, associated with giving some declarative knowledge away
3. *hint*, associated with reminding the student of some fact

4. *acknowledge*, associated with four forms of informing students of the quality of their answer

CIRCSIM-Tutor uses five types of single or multiturn hints, which are positioned on a passive-active continuum, depending on the effort required by the student to find the targeted answer after the hint has been provided. The most active hint is *point to information*, which alludes to some information needed. The most passive one is an *explanation* of the required answer.

Hints are related to the three meta-concepts that are defined for every variable in the concept map being taught. A procedure selects and returns a list of hint categories. Both the types of feedback and their choice model observations from a corpus on human-to-human tutoring in the domain and do not claim to implement a learning theory or pedagogical model. A tutoring history is kept to avoid giving the same hint twice and returning to the same discourse task that has been dealt with already. There was an original modest attempt to separate between the content and the form of hints, however the hint continuum collapses this distinction.

The realisation of hints is template-based for pre-written NL phrases. The variables in the templates are domain-specific and their values are filled in for each task by consulting the concept map. The domain concept map is first searched for availability of content to fill in for each of the hints, until the search satisfies the content of one of the hints in the list. This is necessary because not all relations and causal links are explicitly represented, so that some hints cannot be realised for all tasks. The text of the tutor output is typically a sequence of an acknowledgement, followed by a hint or by giving the answer away, followed by the question addressing the next sub-task. There is some possibility of combining these separate phrases with a comma and of leaving out an acknowledgement to vary this structure.

On the whole, domain knowledge representation is flexible, although not motivated by any cognitive theory, leaving the question of cognitive soundness open. Furthermore, dialogue and tutoring management are not separated and feedback tactics are blended in with discourse plans. As far as hints are concerned, only two dimensions of possible functions are considered: active vs. passive, and content. Therefore, although there is some room for adaptivity, this is also constrained by these limitations.

#### 1.2.1.3 WHY-ATLAS

WHY-ATLAS [Jordan and VanLehn, 2002] is a dialogue-based system that models tutorial dialogues for the domain of qualitative physics. It is based on finite state technology. It uses the concept of *recipes* to implement tutoring strategies as discourse tasks. Recipes are a kind of AI plan, which are authored by domain experts for specific problems. The different kinds of recipes are loaded based on the evaluation of an essay, which the student submits. They can be intertwined, but typically only the remediation recipe is called within other recipes. States in the finite state network correspond to primitives, which produce tutor feedback and represent tutoring goals (e.g. to give an explanation) together with their NL realisations for the specific problem. Arcs correspond to correct student input, which requires no feedback, and pushes correspond to incorrect student responses, which require feedback. Pushes call recipes. Authors define the sequence of primitives and arcs, and specify which recipe should be called from each arc.

Due to the large cost of pre-authored feedback for each problem, Jordan and colleagues [Jordan *et al.*, 2005] turned to the solution of making it easy for non-programmers to author their own course content. They introduced a scripting language, which authors use to specify recipes. The student's essay is evaluated and a recipe is loaded to correct any mistakes. This is repeated until there are no flaws to be corrected. To make it possible to connect more than one output with one goal, authors are required to label tutor dialogue moves that have similar content and therefore may serve the same goal. Authors additionally provide optional steps for each recipe, and difficulty levels for recipes and primitives in order to avoid redundancy. Primitives correspond to dialogue moves that define the degree of difficulty of the tutor feedback, that is, how much effort is left to the student by the feedback. Student dialogue moves are also labelled for similarity of content, which together with the dialogue history maintained by the system, serves as a kind of student modelling and can be used for adaptation. The dialogue history keeps track of the student actions, which are the only ones available to the dialogue manager. When a student dialogue move is in the dialogue history, the current student move shares the same semantic label with it, and the tutor response to it has been labelled as an optional step, the dialogue manager can skip this step. Repetition is thus prohibited. Moreover, if a primitive that is associated with a dialogue move is in the dialogue history and the student has responded to that dialogue move correctly, then the next time the same semantic label appears, a more difficult dialogue move is chosen as tutor feedback. For instance, a "why" question, is followed by a "how" question. There is some evidence that reducing redundancy makes the system more effective.

Recipes merge discourse plans and tutoring strategies. Some flexibility exists on the choice of tutoring strategy. However, the sequences, the appropriate function of feedback, and the NL realisations are all pre-authored, which confines adaptivity and raises the development cost. Discourse flexibility is also not a highlight of the system.

#### 1.2.1.4 PROPL

PROPL [Lane and VanLehn, 2005] is a dialogue-based system that elicits goal decomposition and abstract plans for programming tasks from the students. It teaches skills implicitly through natural language tutoring. It uses expert-like programming knowledge, represented in reusable "chunks" of solution paths. These are called *schemas* and their instantiations for specific problems are called *plans*. Two kinds of skills are taught. Decomposition, which skills involve identifying programme goals and suitable schemas, and composition skills, which involve implementing and integrating the correct plans.

Lane and VanLehn [Lane and VanLehn, 2005] claim that the difference between novices and experts is that novices do not have a library of schemas yet available to them. Therefore, PROPL aims at teaching schemas, as a means of promoting expert-like skills. It uses a three-step feedback that aims to elicit the goal, then the schema for achieving it, and finally the specific plan. The feedback consists of schema steps pre-authored for every programming task and posted as notes for the students once a goal has been identified. The plan is elicited through NL dialogue, which is structured in pre-authored problem-specific tutoring tactics in the form of *Knowledge Construction Dialogues* (KCD). KCDs are chunks of dialogue that concentrate on a main task topic and bottom-up utterances that give away the answer. Tutoring tactics include eliciting hypothetical situations and abstractions, and setting situations for discussion via concrete examples. KCDs and bottom-up feedback can be intertwined. KCDs are an evolution from the Atlas Planning Engine(APE) [Freedman, 2000], which was an attempt to introduce mixed initiative dialogue to Atlas. KCDs were introduced when APE proved to be too expensive, because an increase in content led to increase in the content-specific operators, which are required by the task-based dialogue management, as well as increase in meaning representation for the NL understanding component [Rosé *et al.*, 2001a].

Thanks to the hierarchical structure of KCDs, which implement a main line of reasoning with many subdialogues, KCDs can allow for flexibility in choosing tutoring tactics for different tutoring situations. The main line of reasoning consists of tutorial goals realised typically by a question, which is associated with a couple of expected answers. Remediation goals are associated with each non-correct expected answer. Each remediation goal may have one or two lines of reasoning, which realise it and take the form of subdialogues. What kind of feedback the subdialogues give depends on the detected problem. For example, they may give an explanation, or rephrase a previous question that the student could not answer the first time, or it may consist of more steps in case the problem itself can be decomposed into subproblems.

A study was conducted using PROPL to test the effect of the added NL capability on conceptual learning [Rosé *et al.*, 2001a]. There were two groups, one used PROPL and a control used a version of it without the NL dialogue. It was found that after intervention the PROPL group were better at composition and worked at a more abstract level of schemas and plans. The same behaviour was not observed for decomposition. Rosé and colleagues [Rosé *et al.*, 2001a] interpret that as the result of the fact that the schema steps were posited in the form of notes on how to go about the problem that students had at their disposal, whereas they should have been elicited from the students. However, they argue against using content-free prompts for eliciting schema steps in order to remediate this problem. The pre-authored feedback used by the system does not allow for the alternative of using domain-content feedback that would promote schema acquisition. This would require more adaptivity than KCDs provide, so that the hint content would fit the current cognitive needs of the student and enable the schema to emerge based on the knowledge structure of the student.

Also, to deal with the complicated goal of schema acquisition, NL flexibility would be necessary where a lot of structure can be mirrored, e.g. through discourse cues, and where students can recognise the connections between different bits and pieces presented to them. This additional structure advances coherence and minimises the students cognitive load, who can concentrate on the task.

#### 1.2.1.5 LeActiveMath

In the framework of the project LeActiveMath, an ITS was built for the domain of symbolic differentiation. An integrated course planner generates learning material for students adaptively [Ullrich *et al.*, 2006; Reiss *et al.*, 2005]. The main educational principle behind the course generation is moderate constructivism, which means that students are advised as to what would be the best course to follow, but they are not forced to follow it. Based on this functionality the system can provide students with guiding hints, called global hints, as to what kind of course material they need to concentrate on. For example, whether they need another exercise to practice new learned concepts. In accordance to the principle of moderate constructivism, it is left to the student to decide to read the hints, take them into account, or ignore them. There is a hinting module [Callaway *et al.*, 2006; Porayska-Pomsta and Mellish, 2007] that produces hints for interactive exercises, which is the relevant part to our work. This module is part of the domain reasoner and provides three sorts of messages for every task representing ideal feedback turns for students of different performance and confidence levels. These messages were derived from the analysis of a corpus of human-to-human tutorial dialogues, and they are task-dependent. Hints are either remedial, referring to an incorrect answer, or pro-active, helping the students move on when they are stuck.

Two functions of tutorial feedback are represented in the system: autonomy and confidence. The first refers to the amount of contribution to the task the student is allowed. The second refers to the degree of motivation the student needs. The tutorial planner in the dialogue manager is responsible for choosing the message to output to the student based on the confidence and performance level of the current student. The tutorial planner also takes the output of the domain reasoner, information about the dialogue context and other parameters of the situation like the confidence and performance level. It then manipulates the level of specificity of the message in deciding which of the three possible messages is the most appropriate in context. It may also chose to give the answer away. The student's degree of autonomy is thus controlled. It also determines the level of approval to include in the message by making it more or less verbose based on the level of confidence.

At the realisation front of the tutoring system, the text generation system BUG generates NL realisations of tutorial feedback [Callaway *et al.*, 2006; Porayska-Pomsta and Mellish, 2007]. BUG specialises in the symbolic differentiation domain. It takes as input dialogue moves enriched with rhetorical relations such as *join* – to indicate that two chunks of language must be joined together – and information about the dialogue context, for instance if the same feedback was produced before. From that it constructs verbalisations for the system tutorial feedback by adding pronouns and discourse markers as appropriate, and transforming dialogue segments into natural prose.

This system implements automatic NL generation of tutorial feedback. The hinting mechanism is similar to ours in that it separates the diagnosis from the tutorial strategy. Nonetheless, the implemented tutorial strategy is not very sophisticated. Although the notions of autonomy and confidence are pedagogically motivated there is no clear and unified tutorial model for generating automatic hints for the interactive exercises. Moreover, the researchers do not base the derivation of their hinting algorithm on a theoretical model or on learning effects in general, but on a metric for short term, turn-to-turn performance. This is an adequate metric for deriving their criteria for NL generation, but not for deriving a hinting strategy. Hints are a means to effecting long term learning gains. Therefore, it can be argued that the purpose of hinting is defeated by basing their automation on such short-term effects. Maybe it is for this reason that the researchers talk of a *remediation* strategy, rather than of a hinting strategy. In addition, there is no analysis of the cognitive functions of hints as such. Finally, there is no clear distinction between the function of hints as opposed to other tutor dialogue moves and of those two to other functions of dialogue moves.

#### 1.2.1.6 Summary

The systems reviewed here have two main limitations. They do not distinguish between:

1. tutoring and dialogue functions of feedback
2. cognitive functions and the dialogue move realisation of hints

This poses restrictions on the flexibility and on the possibility of automating the feedback that affect both the tutoring and the dialogue aspects. We propose to address these points by defining a taxonomy of dialogue moves for tutorial dialogues that includes a multidimensional taxonomy of hints. We capture the distinct tutoring and dialogue functions of dialogue moves and the cognitive and dialogue functions of hints in the different dimensions. We also define a pedagogical model and a teaching strategy that realises it whose main unit of feedback are dialogue moves and hints.

### 1.2.2 ITSs with Tutoring Strategies

#### 1.2.2.1 Andes

Andes is a system that teaches college-level students introductory physics [VanLehn *et al.*, 2005]. The goal of Andes is that tutoring pedagogical feedback is separated from content, so that content can be authored by expert teachers. Authoring involves formally representing a problem, checking if Andes can find all possible proofs (i.e. solutions) for the problem, and if the hints provided automatically by the system make sense. Any spotted failure leads to revision in the knowledge base of Andes. The student input is a whole derivation of the solution to a task, which includes forms for defining variables, drawing vectors, writing equations, etc. The proof generated by Andes for the task is also an attempt to model the reasoning behind the student's input. Desirable student reasoning and responses, which are used for evaluating the input, are defined based on extensive cognitive task analysis and are represented formally for each problem. This makes evaluation expensive.

Andes provides feedback on the correctness of the student input. This feedback is two-value: red for wrong, and green for correct. Correct are all answers that are valid. For wrong answers, students can require clarifying feedback. Andes also generates feedback on request when a conceptual error is diagnosed, and unsolicited help for careless mistakes. There is no unified learning theory behind the hints, as Andes does not abide by any, but hopes to facilitate implicit learning of expert-like heuristics. The philosophy of Andes is constraining students' reasoning as little as possible so as to increase transfer of knowledge. There are some exceptions to this rule, which serve efficient learning like requiring that students define the variables that they use.

Unsolicited conceptual hints are normally not provided. First, because Andes does not have a within-session student model, which does not allow reasoning about which hint would be appropriate. Second, because the hints provided are specific to the equation attempted, but Andes cannot reason about which equation the student meant from the input equation. Therefore, the feedback mechanism works as follows. When students request help, Andes asks them which equation principle they are trying to apply in order to deduce from that the correct equation. As a consequence, errors are not always remediated, but when they are, high quality remediation hints are generated. In general, hints are designed to help students figure out the main relevant principles and definitions, and apply them. They are typically presented in a pre-defined order, the *hint sequence*, from ones that lead students less to ones that give away the correct answer. The hint sequence starts either with a prompt to self-explain the next step in the task, or to correct an error in the student's input by pointing out the general problematic area. The second hint directs students to the relevant piece of knowledge. The third hint tells students exactly what they are expected to write. Hints on request can be generated to help students with the next step. In this case, Andes chooses the step and the hints ask the students to identify the main principle involved. Next steps are also associated with a problem solving method, which is a kind of general method for solving similar problems. Because Andes provides solicited hints, help "abuse" is reported, which is the tendency of students to figure out what the hint sequence is and to require a hint by keep clicking on a button until they get the correct answer. To overcome this problem and force students to read hints, Andes hints are presented in a window that freezes and points are deducted from a score total for every answer that the system gives away. This discourages students from requesting too many hints.

An integration of Andes with the Atlas system [Rosé *et al.*, 2001a], was an attempt to allow Andes to pursue its pedagogical goals for some of the problems that Andes can diagnose via natural language dialogue. Knowledge Construction Dialogues (KCDs) [Freedman, 2000], which we described in Section 1.2.1 for PROPL system, were used to this end. Inside Andes, KCDs provide the possibility to give some unsolicited help in the context of a directed line of reasoning, which leads the student through a dialogue specialising in treating a specific problem typical for the task. This is diagnosed based on the pre-specified non-correct expected answers for the task. In order to provide unsolicited help, recognising which step the student is attempting to perform is necessary so as to provide help relevant to it. Conceptual Helper is a tool within Andes that searches the solution graph produced for a task and compares the student action with an action in the graph. KCDs, however, do not handle NL dialogue interactions when students have a problem with equations, or they simply do not know what to do next.

An evaluation of Andes showed that students using Andes were better at the aspects that Andes explicitly aims at tutoring, such as recognising main principles, but not in any other aspects.

A comparative study [Ringenberg and VanLehn, 2006] between two versions of the Andes system showed no significant results between using the standard Andes hints vs. worked-out examples in place of hints. They were both provided on student's request. The post-test asked students to identify similar tasks among a number of tasks. Similarity referred to equivalent underlying domain principles that have to be applied to solve the task. Only near transfer was tested. An interesting result was that the students in the worked-examples intervention chose to do significantly less training problems before they felt they had mastered the tasks. This lead the researchers to suggest ways of integrating hints and worked examples, rather than using one or the other.

Another empirical study [VanLehn *et al.*, 2004] examined the difference between teaching problem-solving strategies explicitly or implicitly. Two groups of participants used two different systems: Andes, for implicit teaching, and Pyrenees, a model-tracing ITS for explicit teaching of problem-solving strategies. The participants had necessary background knowledge in math for the physics tasks assigned, but had taken no college physics courses. The results showed no significant difference in pre-test vs. post-test, when only the correct answers were considered. However, when the same tests were scored based on the progress of the work that led to the final answers, the group that was taught strategies explicitly did better, although they did not reach the required correct answers. In particular, they were able to produce lines of reasoning that included the strategies that they had earlier been taught. This means that there was progress in the reasoning applied by the students and that Andes at least partially succeeds in teaching the lines of reasoning that are implemented in it, but this is not enough to effect correct answers. Such results might be an indication that the students who use Andes do not really acquire a schema that would be necessary for exhibiting the required transfer of knowledge, although they learn an imposed schema, namely the lines of reasoning taught by Andes.

In conclusion, feedback in Andes is only learner-adaptive to a very limited extent. The limited adaptivity creates artifacts like help "abuse", as the hint tactics are very predictable. As both the student modelling and the feedback are designed for every specific problem, feedback is also resource-intensive. Still, the huge development effort that this requires does not produce the required learning effects. This may also be a consequence of the fact that no unified learning theory and analysis of hints are implemented. Cognitive task analysis may be necessary for accessing the cognitive difficulties of a task, but it does not translate into a theory of feedback.

#### 1.2.2.2 Cognitive Tutors

Cognitive tutors is a group of ITSs that are based on the ACT theory [Anderson, 1993]. The main principle of the ACT is that learning is the transformation of declarative knowledge (e.g. definitions) into procedural knowledge (e.g. proving) represented as production rules. Cognitive tutors use production rules in an attempt to model the architecture of human cognition as perceived by the ACT theory. Production rules are heuristics for the applicability of a rule of inference to the current situation, which can also provide the conclusion that should be drawn from the application of the rule of inference. They can be abstractions for the application of various rules of inference, therefore there is no one-to-one correspondence between production rules and rules of inference. Rather rules are clustered together based on common characteristics. Production rules are employed in creating a model of the ideal student, i.e. a domain expert. Moreover, special production rules, called “buggy” rules are also used to model sub-optimal behaviour and incorrect behaviour. Specific production rules for each problem are written, based on an analysis of expert actions when working on the cognitive task involved. Although the student model works on the basis of modelling what a student knows, Anderson and colleagues [Anderson *et al.*, 1993] recognise that there is probably little overlap between the production rules represented in their system and the way students represents the same knowledge, which also varies from student to student.

Cognitive tutors also implement *mastery learning*, a theory that states that complex skills can be learned better when broken down into their components. The part of the cognitive model of the tutors that is called *knowledge tracing* estimates the probability that these components have been mastered, and assists the choice of appropriate tasks to help students master all skill components. The part of the cognitive model that is called *model tracing* matches the student’s attempts to ones that can be generated by the model of the expert student. Thus it monitors the student’s progress. When the student’s actions deviate from those within the range of the ideal model, the system assumes that the student has not mastered the relevant piece of knowledge. Buggy production rules are associated with specific hints, which are provided when such rules are diagnosed [Anderson *et al.*, 1995].

**Geometry Tutor** Geometry Tutor [Anderson *et al.*, 1993] is a cognitive tutor [Anderson *et al.*, 1995] that teaches Euclidean Geometry proofs. The tutor tries to elicit a proof by breaking it down into various steps of inference. Each step of inference consists of premises, a rule of inference, and a conclusion. Students are prompted to input the premises for a step, by choosing from a list of provided statements. Then they are prompted to type in first the rule of inference to which the premises belong, and then the conclusion that follows from the rule.

The general feedback in Geometry Tutor is based on eight tutoring principles including minimising working memory load and other principles of cognition. Nevertheless, there is no analysis of hints and hints in the system are semi-automatic. They are short template-based messages that may explain why a student action is wrong, or what the next action should be. Hints are always delivered in the same pre-defined order and only produced on demand. This creates the problem of having to convince or tutor the students to request help. Moreover, hints are composed for the production rule and the bug that it is meant to remediate. Such "buggy" production rules are written for every cognitive aspect of a task along with the correct rules. Help in Geometry Tutor is in consequence expensive. The specific content of a hint is inserted in the template for each individual task. A similar template-based approach would not work for longer human-like NL messages, which is not among the aims of the system.

**PAT** PAT [Koedinger and Anderson, 1997] is a very successful cognitive tutor, which was developed in collaboration between educators responsible for designing an algebra curriculum for schools in Pittsburgh and the Carnegie Mellon research group. The main objective was to produce a curriculum, which combines the analysis of real-world problems tailored to high-school students in Pittsburgh with the use of computational tools. The goal is that students acquire algebraic skills that they need in real life. The tutor provides an organised curriculum of problems. To help students solve the problems, it also provides tools like spreadsheets, graphs and symbolic calculators, which the students are required to use in analysing the problems, and an Equation Solver, which assists students with solving the equations they create during the analysis.

Feedback in PAT is a mixture of immediate and on-request feedback. Flag feedback is always given for errors, and an accompanying explanation is provided if the error is included in the cognitive model. Misconceptions and common slips are also always treated, this time though by explaining where the problem lies, and suggesting an alternative. The choice of hints that suggest a next activity makes use of production rules. The rules provide information on the students' focus of activity, the status of their solution, and on interdependencies between different problem aspects. A standard sequence of hints is used in connection with each problem aspect. Initial hints simply point students to what is relevant for consideration based on the interdependencies. For instance, the cost of renting from Avis is relevant in calculating the driving distance. The next hint points students to how cost is relevant by asking them to calculate the distance given a specific budget and a rental cost, without providing the exact equation (budget/rental cost = distance). The more requests for hints a student makes on one aspect of the solution, the more specific the hints become. Hints are produced in a pre-determined order.

The system was tested in vivo in three high-schools in the city of Pittsburgh with almost 500 students. In the experimental condition there were two independent variables, the new curriculum using real world problems, and the use of PAT in 25 out of 180 hours of classes. The second condition did not use either of those, but continued with the traditional curriculum. Two skills were assessed, using standardised tests, such as Math SAT (Standard Aptitude Test), which is a standard mathematics test in the U.S. that uses multiple-choice questions for problem solving tasks. First, the students' competence in investigating the real world algebra problems, and second, their competence in translating the problems between the various representations supported by the system. The experimental condition scored 1 standard deviation better on the competencies that the new curriculum targeted, and a bit better on standard algebraic skills. Teachers particularly liked that they had more time to work with weaker students and provide individualised feedback on top of PAT's feedback.

This is indeed a unique effort and its results particularly impressive for the ITS community, since the PAT tutor, and its descendants, are widely used in U.S. schools. Still, there is room for improvement as far as help messages are concerned. The underlying help mechanism is very similar to that of Geometry Tutor and the system is faced with the same problems of high development cost and of having to get the students to use the available on-request help. Also, human-like natural language messages are not provided.

#### 1.2.2.3 SQL-Tutor

SQL-Tutor [Mitrovic and Ohlsson, 1999] and its Web-enabled equivalent SQTL-web [Mitrovic, 2003] are constraint-based tutors. They teach query formation to databases using the SQL language. They maintain a student model that makes use of constraints for reaching correct solutions. A constraint consists of two clauses, the relevance and the satisfaction condition. The general format is *"If <relevance condition> is true, then <satisfaction condition> had better also be true, otherwise something has gone wrong"*. The constraints are manually defined by the designers of the system based on classroom experience for each subdomain. The educational goal is that students acquire *evaluative knowledge*, which allows them to monitor and correct their mistakes. Therefore, the constraints represented do not constitute ideal solutions. Any solution that does not contradict the constraints is accepted.

The student model keeps track of correct applications or violations of the constraints, as well as the tasks that require these constraints as part of their solution. It is a loose student model, which is selective as to the kind of knowledge it represents for pedagogical purposes and does not pick on every mistake. A curriculum is produced for individual students, leading them from simpler to more complex knowledge units, which are represented by the constraints. Multiple solutions to a problem are accepted as correct due to the nature of constrained-based modelling, provided that they meet the constraints.

A pre-determined sequence of feedback is used in principle. However, since student initiative is allowed and feedback is associated with steps, when the sequence of steps is altered by the student, feedback is produced according to this new sequence. Students' attempts are not evaluated step-by-step, but only after the student has submitted a solution. Submissions may also be empty. Upon submission, the tutor provides information on how many mistakes the student committed in total and the place where they occurred. Moreover, the tutor provides hints in the form of general and more detailed descriptions of the error, a correction for every individual error, and the ideal solution to the problem for the student to consult. However, more particular feedback is provided only on one of the committed mistakes, every time the student fails to find the correct answer. A maximum of five attempts to give the right answer is allowed. Students can choose to be shown part or whole of the solution if they wish to. A pre-composed hint in NL is associated with each constraint and this hint is provided every time the same constraint is violated. Students also have unrestricted access to descriptions of databases, tables, attributes, and other basic tools for forming their queries, so that they can abstract from the details and concentrate on the conceptualisation of the queries.

A series of non-controlled evaluations used senior computer science students who were taking an SQL course at the University of Canterbury in New Zealand. Volunteers were recruited for the experimental condition and the rest of the course participants were used as the comparison. The comparison group were by default not using the system. The amount of training was not controlled but involved a minimum of 2 hours using SQL-tutor. Results showed that the experimental condition did significantly better than their course mates in a post-test. Moreover, they mastered individual constraints depending on exposition to learning opportunities of these constraints. This means that the more they solved problems that included such constraints, the better they learned them. The experimental condition also produced a learning curve in compliance with the power law, which says that learning starts fast and stabilises at a high point over time.

Two great advantages of constraint-based tutors are the effective knowledge representation and the student modelling. Evaluating student solutions based on constraints saves the effort of having to generate all possible solutions. However, there is no guarantee that this is not an artifact of the existing structure of domains like posing SQL queries already. Most importantly, SQL-Tutor is missing an underlying learning theory, which results in ad hoc feedback. This feedback is nonetheless expensive, because it has to be pre-defined for each constraint.

#### 1.2.2.4 Sherlock

Sherlock [Lesgold *et al.*, 1992] and its follow-up Sherlock 2 [Katz *et al.*, 1998] simulate an electronics troubleshooting environment for malfunctioning aircraft electronic modules. It is used by technicians who practise how to trace the source of the problems detected by the F-15 Manual Avionics Test Station.

The goal is to make the environment as close to the original as possible so as to model situated learning, that is, learning in its natural environment and learning opportunities. It provides an abundance of problem situations for practice in a short amount of time that otherwise would take years for every trainee to encounter. It has been shown to effect the same level of expertise with 20-25 hours of training as 4 years of actual experience.

Feedback is mostly given on request with two exceptions: Hints that aim at preventing students from continuing on an obviously wrong solution path, and hints that are produced after redundant actions to inform the trainee of the redundancy. When help is requested, the system offers a planning menu, which requires the student to step back, look at the actions taken, and re-think the overall strategy for attacking the problem. Otherwise, feedback is provided in the form of hints. The troubleshooting routine consists of four abstract phases, which are ordered and define its structure. When all levels of hints in one phase have been exhausted, the system provides hints for the next phase. The appropriate hint to produce includes two decisions: (i) which of the four abstract phases of troubleshooting to address, and (ii) how much information to reveal following a 1-5 scale from less to more information. The first decision depends on which phases the student has completed already. The second decision depends on an estimate of the expected performance of the student at the point in troubleshooting. In the hint scale, level-1 hints are a dynamic recapitulation of the actions the student has taken so far. Levels 2-5 consist of hints composed by experts for every problem solution in a solution space, which are explicitly represented. So, although level-1 hints are automatically produced, the overall hint cost of the system rises significantly with every task and every solution that have to be represented, and due to the addition of hints by experts for each of the solutions.

#### 1.2.2.5 Why System

The Why System [Stevens and Collins, 1977] is a system that implements a Socratic strategy in NL dialogue. It teaches reasoning about the factors of rainfall in different regions.

The domain theory is implemented by production rules of the form “If in situation X, do Y” to express it in a procedural manner and abstract from the content of specific problems. Expert knowledge is further organised in scripts and subscripts under the assumption that this is how knowledge is indeed organised in human cognition. Scripts and subscripts represent temporal steps and causal factors concerning rainfall. These define the tutorial goals for the problem and are used to lead the dialogue on the problem, which is represented by a script. This script essentially goes through the steps and factors and tries to elicit them. Subscripts decompose the scripts and represent tutorial subgoals. They are evoked whenever the student has a problem with one of the goals of the primary script in order to resolve the problem. Tutorial goals are dependent on the structure of the knowledge that has been taught. A semantic grammar and a matching procedure try to interpret the student’s input and recognise such steps and factors in it. Natural language understanding in the system can handle only very simple NL input, as it does not deal with anaphora or conjunction.

The Why System concentrates on the characteristic of the Socratic method known as entrapment, which challenges students to realise the faults in their reasoning. The general tutorial goals of the Why System are to refine the student's causal model, reasoning, predictive ability, and self-correction of bugs in the student's knowledge. Based on these goals, the system chooses examples to tutor, which exemplify the knowledge that the system wants to teach. It models tutoring techniques, which characterise observed human tutoring.

The kind of bugs that may occur in the student's input and that the system considers are categorised in five classes:

1. factual bugs
2. severe misconceptions that are not directly relevant to the tutorial goal
3. overgeneralisation of situations
4. overdifferentiation of situations
5. reasoning strategy bugs

Bugs of categories 1 and 2 are dealt with by simply correcting the student. For categories 3 and 4 the tutor uses counterexamples to demonstrate that students have not used sufficient factors or that they have used superfluous factors. Treating category 5 bugs depends on the specific problematic reasoning strategy. For example, to teach how to form a hypothesis, the tutor asks students to look at the relevant factors and tries to lead them to extract patterns that can form a hypothesis [Collins and Stevens, 1991].

This is a system with elaborate teaching tactics and an advanced representation of tutorial goals. What it is lacking is an analysis of feedback at the bottom level that realises these tutorial goals of the kind hints and dialogue moves can provide.

#### 1.2.2.6 The MDP Approach

The MDP (Markov decision processes) approach [Barnes and Stamper, 2008] is an attempt to use educational data mining to create student models and provide hints. The original idea started from making use of the cognitive tutors model and learning from previous data to automatically generate production rules for problems [Stamper, 2007]. The feedback associated with these rules could then be provided. MDPs, which is a reinforcement learning technique, are used to learn from passed student data. The probability of the next step that the student should attempt is estimated and based on this next step hints are automatically generated. Each proof attempt is represented as a graph with a sequence of states (the premises produced after a student attempt) representing the partial solution up to each point. Student actions (the application of rules to the premises) connect the different states. The solution graphs of all passed student attempts are combined into a single graph to represent the space of all solution paths followed previously. Reinforcement learning is then applied to find an optimal solution to the MDP.

Hinting with this method is not adaptive, but the following sequence of four on-request hints is generated for each step:

1. indicate a goal expression to derive
2. indicate the rule to apply next
3. indicate the premises (lines) where the rule can be used and
4. bottom-out hint combining 1–3

These hints are contextualised and aim at helping the student to concentrate on an appropriate next sub-goal. They are verbalised by experts once and the canned verbalisations are produced every time the particular hint appears on the screen of a student.

Hint Factory, which uses the MDP approach, is an extension that was added to the system Deep Thought to provide automatic contextualised hints for deductive logic [Barnes *et al.*, 2008]. Hint Factory comprises an offline MDP generator and an hint provider, which produces hint files for each step online. Hint Factory's ability to generate online hints was tested by using the system in a class of 40 on four problems in 2008. MDPs were generated automatically from processed data of the 2007 class that used Deep Thought without Hint Factory.

The most interesting of the results reported is that a hint could have been produced 48% of the time if the students had requested a hint after every attempt. However, the system did produce a hint in 91% of the cases when the students did actually hit the hint button. Barnes and colleagues [Barnes *et al.*, 2008] conjecture that there might be some pattern in the cases when the students request a hint that would coincide with special difficulties in the domain. Unfortunately, no data is available on the use and effect of hints over time.

LeActiveMath

(From D24) The goal of the LeActiveMath project is to design an intelligent Web-based e-learning system for mathematics. One of its major features is adaptivity. The mathematical course material presented to learners is adapted to the learners' goals, the learning scenario, the learners' competency-level, and their individual preferences. Furthermore, LeActiveMath integrates tools that can be used in exercises and for exploratory learning. The Tutorial Component is the central component for the adaptivity of LeActiveMath and the provision of moderate constructivist learning opportunities. The adaptive features are motivated by pedagogical and cognitive research. In particular, LeActiveMath is learner-centered and supports the learners initiative. The technology as well as the content that is developed to evaluate the technology realizes a moderate constructivist and problem-based approach to learning and teaching mathematics.

The aim of LeActiveMath is that students become autonomous learners, but wants to support weak students by organising their course material for them to facilitate access to content. Pedagogical strategies help students develop learning paths through the content. (This is again at the macro level. Strategies refer to strategies for organising the learning material, rather than strategies for attacking problems in an interactive session, to the effect that the LAM strategy would tell the student to do an interactive exercise. It is more meta-cognitive. A strategy assists students at organising their courses, where each course consists of a sequence of learning objects like Exercise, Simulation,Questionnaire, Diagram, Figure, Self-Assessment etc. `Menon` can be seen as a submodule to be support the *Exercise* learning object.) For instance, the LearnNew strategy (cite D20) would include introducing the new topic, teach the new concepts, pracise via exercises, make any connections to known material and reflect on what was learned.

The Exercise Sequencer within LeActiveMath is the interactive task module, which chooses the sequences of tasks and presents feedback on individual tasks. Competencies are used in the determination of the exercise sequence. The aim of the Exercise Sequencer is to lead the learner to the next competency level. The Suggestion Agent is responsible for providing local and global feedback on the student's activities in solving an exercise. Global feedback suggests ways of consulting the learning material. The local feedback, which is relevant to our work, provides feedback on individual exercises.

The architecture of the LeActiveMath server for providing global feedback includes various components. The Action Capturing component informs the server about the student's actions. The Basic Diagnoser analyses the various observable values of the student actions, included an assessement of level of competency. The Diagnostic Reasoner is responsible for diagnosing non-observable problems. The Suggestion Reasoner can provide one of the following suggestion objects: a summary, a feedback message or a speech act to be realised in NL. This approach is similar to ours in that the presentation of the suggestion objects is not pre-determined, but is send for realisation to the Suggestion Renderer. Menon goes beyond this approach in that it formalises hints in terms of dialogue moves and hence allows for dialogue and NL adaptivity in the local feedback.

The tutorial component has to provide an interface for the tutorial dialog components. As we learned from our previous experience, this is a non-trivial task. Among the difficult research problems are: (1) react to diagnoses of the learners activities and state may come from different As we learned from our previous experience, this is a non-trivial task. Among the difficult research problems are: (1) react to diagnoses of the learners activities and state may come from different.

Requirement 4.12: The tutorial component (TC) allows to represent and execute different pedagogical strategies, e.g. problem-based, traditional didactical approach. Supports Different ways of teaching/learning. Because Different ways of teaching use different pedagogical strategies. Check-rule Empirical test with small groups using different strategies.

They implement different pedagogical strategies. An ontology provides the instructional purpose of learning materials in order to allow re-usability of content and pedagogical knowledge and is used for the implementation of the multiple pedagogical strategies. This is the macro level of our ontology for hints.

The tutorial component generates symbolic representations of transitions, introductions, and summarisations to be verbalised into English by the NLG component in order to avoid boring the students with canned text.

The pedagogical strategies and the adaptive content selections support the notion of competencies in the sense of the Program for International Student Assessment (PISA) study. A competence may be seen as the observable application of knowledge that allows a student to perform a task. Acquiring different competencies makes up mathematical literacy. The formalised pedagogical strategies are a basis for content selection accordking to the student model. The aim is to capture information about the student's current knowledge state, preferences, learning history and style, misconceptions etc. However, there is no ontology to support switching between different learner models with varied representations of individual information. Our ontology uses the abstract representation of hint content in terms of instructional points that are in turn used by the the session model (HSS).

Requirement 4.17 The TC offers scenarios that target competencies. Supports Competency-level pedagogy as used in PISA and other studies.

An evaluation of the automaticc course generation vs. pre-generated courses that included 11 subjects yielded moderately positive results for liking the system more. There was no between-groups difference in the assessment of the usefulness of the system. A lab study where 11 subjects took place and did not include the Tutorial Component gave overall positive results, with some complaints about complexity of the system for. Results also suggested that students were expecting to see at least some of the functionalities that the Tutorial Component provides, like suggesting exercises based on the student's knowledge.

#### 1.2.2.7 Summary

These are the main drawbacks of these systems that we propose to address. Some or all of these drawbacks apply to each of the systems.

1. limited adaptivity of pedagogical feedback for different students
2. limited adaptivity of NL realisation of feedback
3. expensive pre-compiled feedback for particular problems and solutions

The analysis of the multiple dimensions of dialogue moves and hints that we undertake makes it possible to model the various underlying functions of them explicitly and relate them to the tutoring and dialogue context dynamically. This adds flexibility at the pedagogical level and opens the road for more flexibility at the NL realisation level. As an alternative to pre-compiled feedback, we define a domain ontology that is inspired by schema theory and defines abstract concepts and relations that serve as instructional points in tutoring. We use these along with other domain-independent parameters to define tutoring situations. The teaching strategy that we implement involves numerous sub-strategies that select the appropriate dialogue moves and hints according to the tutoring situation. The domain content of the chosen hint is specified by such abstract instructional points, which can be instantiated for arbitrary problems and solutions in a domain.

### 1.2.3 ITSs with Tutorial Dialogue Management

#### 1.2.3.1 STEVE

STEVE [Rickel *et al.*, 2000] is an evolved version of a series of tutors built by the same group, which are all domain-independent task-oriented dialogue systems. STEVE uses a pedagogical agent that cohabits a virtual environment and teaches the operation of complex equipment like air compressors. It supports both actions and utterance input by students, but no natural language. It can demonstrate actions in the environment, gaze and gesture to capture and direct the student's attention. It explicitely represents procedural tasks in terms of steps with ordering constraints, causal links, and end goals.

This mixed-initiative dialogue system implements SharedPlans [Grosz and Sidner, 1986; Rich and Sidner, 1998], a model of planning in task-oriented dialogue, which centralises the goal of the task and emphasises the collaboration between the collocutors for achieving a common plan at each time during discourse. As a consequence, when students interact with STEVE they can propose a goal, debate who should perform the goal and how it could be achieved, and can discuss the values of the parameters of a goal. STEVE provides explanations to students in the form of text. The system learns the text for the explanations by running the procedures representation on specific problems and producing the solutions to them.

Within the framework of STEVE [Rickel *et al.*, 2000], Diligent [Angros *et al.*, 2002] is a tool for learning domain procedural knowledge and providing respective explanations in natural language. It acquires knowledge by observing an expert's performance of a task, as a first step and subsequently conducting self-experimentation. Finally, human authors correct what Diligent has taught itself.

The only kind of hints STEVE delivers are suggestions on the next step of the task. Causal links are used for automatically generating explanations of how the suggested steps contribute to the task. Hints for helping the student to come up with the next step are not delivered.

#### 1.2.3.2 BEETLE

BEETLE [Core *et al.*, 2000; 2001; Zinn *et al.*, 2003] is a system that teaches electronics and electricity implementing a constructivist model and NL dialogue. It uses conversation games, which is a kind of recipe for managing dialogue situations. Emphasis is put on managing tutorial dialogue. An architecture is proposed where the knowledge sources of pedagogical strategies, domain knowledge, and dialogue management strategies are clearly separated [Zinn, 2002]. The architecture is based on the *information state* approach to dialogue, which is a data structure that can be dynamically updated with pre- and post-conditions for performing dialogue moves and other relevant domain-dependent and domain-independent information for managing dialogue, like a dialogue history [Bohlin *et al.*, 1999]. The architecture comprises three modules: (i) *Interpretation* analyses the student input with respect to its content, the dialogue acts it performs, and its contribution to the task. (ii) *Update* is responsible for planning the next dialogue move to perform at the dialogue level. (iii) *Response generation* chooses appropriate tutorial feedback and generates the NL realisation of it, adding other relevant dialogue aspects.

The Response Generation module uses a 3-tier planning. The top tier composes abstract plans that consist of discourse tasks, including tutorial feedback, which should be performed next. It writes the discourse plans into a *task agenda*. It is activated when the tutor agent is engaged in a tutorial dialogue to create an initial abstract plan, or later to repair a plan on the fly if it fails. It also monitors the satisfaction of discourse obligations, which is the discourse theory implemented in the system. The middle tier groups and describes ways for performing the tasks in the agenda in different dialogue situations, which translates into specifying dialogue acts to be performed. The bottom tier gets a sequence of dialogue acts and compiles a natural language utterance that realises them. This architecture makes the system re-configurable for different domains, dialogue, and pedagogical theories.

The system implements directed lines of reasoning that are abstract domain-independent tutoring tactics. Directed lines of reasoning are chosen based on an agenda of unachieved tutoring goals and dependent on fulfilling constraints and preconditions that are associated with them. They are associated with specific conversational games. Conversational games consist of particular dialogue moves that realise directed lines of reasoning at the dialogue level. No hints as such are implemented, although the dialogue moves involved in conversational games do include some hint functions, like prompting the student.

The direction was changed towards more research on tutoring feedback in the BEETLE2 tutorial dialogue system [Dzikovska *et al.*, 2008]. BEETLE2 uses a tutorial planner that defines classes of dialogue acts, like accept. These dialogue acts can be specified further and they can be verbalised in different ways depending on tutoring context. A deep generation system is responsible for the verbalisation. It uses deep syntactic structures and lexical items. This architecture is used to implement a tutoring strategy that adapts to the diagnosis. A corpus of human-human tutoring on D.C. circuits was collected transcribing tutorial dialogues between 30 students taught by 3 expert tutors. The corpus was analysed to derive first basic guidelines on an appropriate tutoring strategy. The diagnoser was built based on this analysis that revealed the importance that correct, incorrect and missing parts are taken into account in the design of tutorial feedback. The diagnoser uses a comparison of the student answer to the three categories of possible correct answers. Depending on the combination of correct, incorrect and missing parts, the tutorial strategy chooses to address any errors, acknowledge the correct parts and prompt for the missing ones, or indicate an irrelevant answer and try to focus the student to the topic of interest. The innovative aspect of this research is that it introduces the notion of different possible correct answers, meaning that the tutor should accept answers even if they are not ideal, but should provide additional feedback on the already correct answer. Therefore, the categories ideal, good, and minimal correct answer are implemented. Depending on the degree of correctness, the system just accepts ideal answers, accepts good answers and repeats the key point, or accepts and rephrases minimal answers to model a better answer.

BEETLE2 is based on the realisation that diagnosis, feedback strategy, dialogue planning and feedback generation should be separate. However, the research concentrates on the verbalisation end of the feedback and not on the cognitive function of hints. The diagnosis technique is a good start for this purpose, but for the purpose of more cognitive and pedagogically oriented feedback the bigger picture of the student model must be integrated. An obvious starting point would be to decide on rephrasing or not a minimal correct answer depending on whether there are reasons to believe that students might be overwhelmed by rephrasing it.

#### 1.2.3.3 Summary

These systems have developed dialogue management and/or generation techniques, but are lacking on the whole elaborate feedback mechanisms and hints in particular. This is the kind of feedback that we investigate in this thesis.

### 1.2.4 ITSs Promoting Self-Explanation

We dedicate a section to systems dealing with self-explanation, as there is a trend in the ITS community of systems that concentrates on this specific aspect of tutoring. The relevance to our work is that these systems make use of hints to promote self-explanation.

#### 1.2.4.1 SE-Coach

SE-Coach [Conati and VanLehn, 1999] was built as a module embedded in Andes to help students learn through examples and self-explanation. The domain is Newtonian Physics and the focus on metacognitive skills. SE-Coach provides wrong vs. correct evaluation feedback. In addition, it uses a probabilistic student model based on current student actions and prior student knowledge to assess problems and remedy them via promoting self-explanation.

The student is provided with a list of "self-explanations" and is required to choose the applicable one each time. Hints help students to monitor their learning at the metacognitive level. "Self-explanations" are pre-formulated for each problem and hints are pre-compiled. Apart from the standard problems this creates, a valid self-critique by the developers is that not only are pre-compiled "self-explanations" inefficient, but also the system can potentially hinder effective self-generated self-explanation. In a sense, pre-compiled "self-explanations" defeat the point of self-explaining, which is to activate deep cognitive processes in the students by requiring that they come up with their own explanation.

#### 1.2.4.2 PACT Geometry Tutor and Geometry Cognitive Tutor

New-generation cognitive tutors have been developed on similar principles. The PACT Geometry Tutor [Aleven *et al.*, 1999] is a cognitive tutor, which teaches high-school geometry with real-world situation problems and is used in US schools. Students working with the PACT Geometry Tutor must calculate the unknown quantities in a problem and must explain their step by choosing a geometry theorem or definition from a provided glossary. The main features of the tutor's feedback are hints on request, which encourage the student to use the glossary in the described way, and the ability to highlight a smaller number of options in the glossary to reduce search. The tutor also uses additional feedback for eliciting self-explanation, which was shown to increase learning gains in a preliminary study (ibid.).

An advanced version of Geometry Cognitive Tutor [Anderson *et al.*, 1993], which elicits self-explanations in NL dialogue, requires students to use their own words to explain their answers. There are two main principles behind the approach of asking students to self-explain in their own words. First, it forces students to recall the relevant pieces of information from memory. Second, it is less likely that students will have problems with the use of jargon and terminology [Aleven *et al.*, 2001]. Feedback in the new system is based on a classification of the explanation that the students provide. Detailed feedback, which does not give away but elicits the explanation, is used. It consists of problem specific questions, which pick up key words from the student's solution and explanation up to the point, and ask the students to state the general rule for the principle behind those key words.

A comparative study between the form of self-explanation used by PACT Geometry Tutor and using one's own words to self-explain showed that when students used their own explanations they did not overall learn better. However, this sort of self-explanation in combination with high-quality feedback gave better results. Moreover, students who did not mention general principles in their self-explanations, but rather specific problem examples learned less [Aleven and Koedinger, 2000a].

As these tutors use the same help mechanisms as all cognitive tutors, the same possible improvements apply. Additionally, the results of the study reveal the importance of feedback as opposed to mere self-explaining. The results also emphasised a problem with menu-based approaches to student input, namely that students are far more likely to guess, rather than try to think. A side-effect of this is that the system is driven away from estimating the actual level of skill of the student, and it is hard to diagnose and remediate problems.

#### 1.2.4.3 Summary

The systems reviewed in this section concentrate on only one of the pedagogical aspects of the feedback that we consider for the definition of our teaching strategy and hints, namely self-explanation. They also use either menu-based approaches or pre-defined hints, which result in limited flexibility on the NL dialogue level and on the tutoring level respectively. We address both of these issues by separating these two levels. We scrutinise the tutoring level and provide the basis for flexibility on the NL dialogue level.

### 1.2.5 ITSs Promoting Motivation

#### 1.2.5.1 Wayang-West

There is a plentitude of work on motivation, but we only mention here that of Beal and colleagues [Beal and Lee, 2005]. Their system, Wayang-West, teaches mathematics to secondary school students based on SAT-Math tests. It is engaged in providing instruction in a motivationally informed manner. Their rationale is that the student who is not motivated will not attend to the instruction. Their system aims at directly increasing motivation and engagement in an adaptive way, due to the fact that personality traits affect if a person is motivated or demotivated by the same feedback.

Wayang-West uses a model that takes such issues into account as (i) self-report on motivation, (ii) indications of lack of motivation, e.g. the student making a habit out of asking for help or requesting all help available in a row, (iii) the cognitive ability of the student, judged on the basis of previous interactions and the knowledge the student should have, and (iv) the kind of help that seems to have an effect on the student's progress.

Student engagement and motivation is promoted by varying the system output depending on the student model. The aim is not to solely avoid frustrating the student by giving unrestrained help, but rather to involve students in the task, in order to let them experience the joy of learning and practice to overcome frustration. The system thus balances providing instruction and maintaining student motivation. This may include providing or refusing help. Help may be refused for example, in case the student's cognitive ability represented by the model indicates that the student should be able to do without help. Motivation is also controlled by reporting progress and by the way progress is reported to the student. Motivational feedback is provided together with progress report, modelling human tutors. Motivational feedback may consist of positive feedback, like *I know you can do it*, or feedback on how much effort is needed. Other aspects of tutoring that can be adapted pertain to adjustments made previous to the interactive session. Such adjustments are no are not dynamic. For instance, the difficulty to the assigned task can be adapted.

This work is a good example for what is involved in taking motivation into account in designing and providing feedback. It highlights what is missed out by systems that do not treat motivation and it points to the necessity of incorporating a motivation aspect to hinting. However, it does not implement automatic multidimensional cognitive feedback, but rather only empahsises the affective aspect.

### 1.2.6 Theoretical Work on Designing Feedback Strategies

In this section we explore the work of *Interactive Two Feedback-Loops Model* or ITFL-model [Narciss, 2003]. Although this is only theoretical work, we give an account of it here because it is the only work to our knowledge that attempts to provide a structured analysis of tutorial feedback. This analysis can be seen as the macrolevel equivalent of tutorial feedback, in general, and to the analysis this thesis undertakes for hints, in particular. It aims to provide guidelines to ITS developers for composing tutorial feedback based on multiple dimensions.

The ITFL-model is a theoretical model for designing tutorial feedback. The model identifies the interaction between an internal and an external feedback loop as the point where learning takes place. The internal loop refers to feedback that processes variables to which the learner has direct access. They relate to prior knowledge, metacognitive skills, and cognitive skills, e.g. perceived effort, self-assessment of skills, and representation of a task. The external loop relates to learning material or instruction. Types of variables processed by the external loop, are the instructional goals, the diagnostic procedures, and the feedback quality. The external feedback loop is the one most relevant to designing tutorial feedback, but it is always dependent on the internal loop due to the prerequisite interaction between them for learning to occur, and especially from the perspective of active or self-regulated learning.

According to the ITFL-model, the distinction between internal and external feedback loops requires certain key processing tools, which may be seen as tools for efficient instruction:

- Sensors, which register the current values of variables. In ITSs, this is normally the job of the student model.
- Reference values, which define desirable values of the variables and are dependent on individual goals (for internal variables), and on tutorial goals (for external variables). In ITSs, this is typically done by defining cognitive skills, or learning components and their respective desirable values.
- Controller, which compares the current values registered by the sensors with the reference values. ITSs typically model an external controller, which compares the current values of the student's cognitive skills or mastery of learning components with the values required by the tutorial goals.

In case of discrepancies between the current and the reference values, the controller is also responsible for producing control actions to remediate the discrepancy, e.g. some instruction. For successful instruction, the control actions are produced based on a well-defined control process, which acts on equally well-defined variables, and specifies how these are measured and regulated. In ITSs this involves identifying the requirements associated with the instructional content, the tutorial goals and the taught tasks. For example, the difficulty of the task must be represented, as instruction should be more elaborate for tasks of medium complexity, but the same degree of elaboration is not needed for easy or for very complex tasks.

The ITFL-model sets specific prerequisites for the external loop, and hence for the design of instruction. Learning goals must be operationalised so that reference values can be defined and used for the evaluation of the success of instruction. Indicators of mastery must be defined in a valid and reliable way. Instruction must be able to transform the discrepancy value into a piece of instruction that provides information highly relevant to the previously set requirements for mastering the task.

Various functions of external feedback currently used in ITSs are identified. These include an acknowledging or reinforcing function, an informing function, a guiding or steering function, a regulatory or correcting function, a motivational function and an instructional function. However, these functions are not specific enough for the purposes of the ITFL-model, as Susanne Narciss argues:

> *Since finer differentiations of feedback functions make it possible to work out which information will be useful in which settings, the careful selection and specification of the intended feedback functions provides the basis for designing tutorial feedback* (p.13)

Therefore, a more detailed analysis of feedback functions is provided by the model. Functions are divided into three major categories with their subcategories:

1. *Cognitive functions.* These relate to errors that may occur, because the learner is lacking some content-related, procedural or strategic knowledge, or because the knowledge is interlinked incorrectly, or because the conditions for using the knowledge are incorrect or ill-defined. Cognitive functions are subdivided into:
    (a) an informative function, which provides the required information in case no further diagnosis of the committed error is possible
    (b) a completion function, which provides lacking knowledge
    (c) a connective function, which provides information about connecting pieces of knowledge
    (d) a differentiation function, which provides information to clarify imprecise knowledge
    (e) and a restructuring function, which provides information on correcting wrongly connected pieces of knowledge
2. *Metacognitive functions.* These refer to external feedback for improving metacognitive skills. They are subdivided into:
    (a) an informative function, for providing feedback about metacognitive strategies

   (b) a specification function, for providing feedback about criteria for monitoring goals or the conditions of application of metacognitive strategies
   (c) a corrective function, for providing corrective feedback on metacognitive strategies
   (d) and a guiding function, for encouraging students' self-regulated learning and the deduction of strategies for self-regulation

3. *Motivational functions.* These functions are normally an integral part of all types of feedback, even if they are not explicit. Sub-functions include:
   (a) an incentive function, which makes the results of processing the task visible
   (b) a task facilitation function, for providing information on overcoming difficulties with the task
   (c) a self-efficacy function, for providing information that helps mastering the task, despite the committed errors and the difficulties with the task
   (d) and a reattribution function, for providing information that allows connecting mastery experiences to personal causation

Finally, a distinction is made at the utterance level between an evaluative and informative function, which according to the model should be the two components of the feedback utterance.

There are two major differences between this model and our work. First, this model only wishes to equip ITS developers with guidelines on the many dimensions that one must take into account when implementing tutorial feedback. It is therefore not detailed enough to be implementable and leaves the specifics of how these dimensions can be used to deliver automatic feedback open. Second, although the model is somewhat aware of the different dialogue function of tutorial feedback, it does not lay out how dialogue and cognitive feedback relate, or how they should be treated separately in ITSs, which is one of the main contributions of our research.

### 1.2.7 Dialogue Systems: Discourse vs. Task Planning

Looking at it from a different but similar perspective to that of BEETLE [Zinn, 2002], tutorial dialogue is a form of task-oriented dialogue [Allen *et al.*, 2001a; 2001b]. The task is to solve a problem collaboratively with the user (student), irrespective of the specifics of how this collaboration should exactly look. This is dependent on the pedagogical model and the tutoring goals of every ITS. ITSs can benefit from existing research that investigates task-oriented dialogues in the dialogue community. Here, we look at an example of a system from that area of research.

TRIPS [Allen *et al.*, 2001a] is a dialogue system where task and discourse planning are clearly separated. The latter needs to be informed by the former in order to interpret the user's utterances, but keeping them separate allows flexibility. TRIPS consists of four main components, which work asynchronously:

1. Interpretation Manager: This interprets the raw input, assigns dialogue moves and task content to the input, which involves recognising the intentions of the collocutors in the domain, and informs the status of the discourse context, which is a separate module.

2. Behavioural Agent: This plans the system's utterances based on its goals, on pending obligations, on the user utterances, and on any changes in the world. For the latter, a world model is consulted, which is kept by the system as a representation of what modifications the users undertake in the physical worlds vis-a-vis the decisions made during their dialogue interaction, i.e. in performing the task.

3. Generation Agent: This receives actions and directives from the behavioural agent and plans their NL realisation based on pending obligations.

4. Task Manager: This uses an abstract problem-solving strategy and is responsible for interpreting what the objects referred to in this abstract solution mean, and the way operations on them happen. It must create a solution for the problem, build a specific course of action from the abstract one, and evaluate the intention recognition behind the dialogue acts that the interpretation manager reads in the user's utterances. Its output is then used by the interpretation manager, the behavioural agent, and the generation manager.

The flexibility that characterises the separation of task (task manager) and discourse planning (behavioural agent) in this architecture also affects the subsequent improvements that the system might need. The two planning levels make it easier to test and detect problems in the levels separately, as well as make adjustments to one level without necessarily having to adjust the other. Most relevant to tutorial dialogues is that this architecture makes incremental processing possible, as not all kinds of input need to be treated uniformly by the dialogue manager. A simple example of this is handling acknowledgements, for example saying "OK" to signal that one has understood the linguistic meaning of an utterance by one's collocutor. Acknowledgements are part of what is called *grounding*, that is the process of making sure that the collocutors agree between them on what has been established so far in the dialogue, which is common practice among humans and very important in task-oriented dialogues in view of the need to collaborate in resolving the common task. The generation manager in TRIPS may generate an acknowledgement based on adjacency pairs, which are dialogue moves that are typically produced one after the other. No problem solving interference needs to be involved in this simple planning. In the meantime, the behavioural agent may still be planning the next actions that require the involvement of the task manager. Moreover, the system may need to plan an acknowledgement at the discourse level, but reject the content of the utterance at the task level. In tutorial dialogues this happens quite often in case of a wrong contribution by the student. The tutor acknowledges the contribution, as for instance the theory of discourse obligations predicts (cf. Chapter 4), but then rejects its content. In consequence, the system can reason about producing an acknowledgement, while the information related to the tutoring task is being processed in parallel.

In ITSs in general, by maintaining different dialogue and task plans, a system may implement a dialogue and discourse theory at the dialogue level, and a distinct learning theory at the task level, thus doing justice to both of these complex issues. As a result, the task plan remains the same even when the dialogue plan, which is responsible for the interaction rather than the performance of the task, needs to change on the fly [Larsson *et al.*, 2002]. The task plan in ITSs refers to the choice of topic or step to teach next and the choice of the feedback strategy to teach it, which depend both on the content of the history of the interaction, e.g. on the student model. The dialogue plan refers to the way the feedback strategy should be expressed in dialogue moves and discourse markers. The next step and the feedback strategy can be determined by the tutoring situation already and may be the same for different dialogue situations. On the contrary, the dialogue plan must be adjusted to the dialogue situation, and hence to the specific previous dialogue move of the student. Note that adaptation is needed on both levels, which makes managing adaptation in tutorial dialogues complicated and requires separate deliberation.

The kind of flexibility reflected in such an architecture and the possibility of adaptation that it facilitates is what is missing from current tutorial dialogue ITS. Such characteristics would enable paying due attention to the separate aspects of tutorial dialogue feedback in developing ITS. The same characteristics would also make it possible to tease apart the different parameters of feedback to test them and give insight into what makes feedback effective.

## 1.3 The Research Problem and our Approach

In this section we define the research problem, describe the general approach to NL tutorial dialogues that is the background of our research, and analyse the specific approach to automatic feedback for NL tutorial dialogues, which is the topic of this thesis.

### 1.3.1 The Research Problem

The insight that natural language is important in tutoring is obvious in early attempts at building NL tutorial dialogue systems like the Why System [Stevens and Collins, 1977]. Notably, however, these research directions did not seem to be aware of the complexity and the varied constituting parts of human tutoring, namely adaptive cognitive content and dialogue/discourse sensitivity. Moreover, these first attempts were also hindered at the time by the immature theories and computational models of dialogue and discourse [Corbett *et al.*, 1997]. In order to circumvent this problem, subsequent approaches to NL tutorial dialogue systems were directed toward simplified models of dialogue management for specific genres and domains (e.g. [Person *et al.*, 2000]). In such approaches, tutorial feedback is inseparable from dialogue phenomena posing limitations on both.

Some twenty years later, NL dialogue/discourse management has matured and dialogue managers have been built that can handle complex genres (e.g. [Core and Allen, 1993; Poesio and Traum, 1998; Cooper *et al.*, 1999; Porayska-Pomsta and Pain, 2000; Allen *et al.*, 2001a; 2001b; Zinn, 2002; Dzikovska *et al.*, 2007; Ferguson and Allen, 2007]). A dialogue manager is the component in a dialogue system where the interaction of the different components in the system is managed, and the dialogue theory is implemented. The decision on which is the best response to the user is based on the dialogue theory. There are different approaches to dialogue management, with the main ones being finite-state based, form-filling, and information state management.

Finite state automata are the simplest way of modelling dialogue management. There is always an initial state. States are normally defined with certain expectations as to what the system can have as input. Different transitions are defined that take the user input into account in order to decide what the new system state is. As soon as a particular input is recognised by the system, a predefined behaviour is produced by the system. That results in the system being set to another state, which is itself predefined. Finite-state based systems are easy to build for simple domains. The increase in complexity of the domain and the dialogue phenomena that need to be modelled lead to an exponential increase of states and transitions to define. This poses limitations to what genres and domains can be modelled with this technology. Therefore, finite-state based dialogue managers are normally used for domains where the system asks questions and the user provides the information required and the set of questions and possible answers is small. An example of such a manager is the CSLU toolkit [McTear, 1998].

Form-filling approaches are a bit more advanced than automata. Slots are used to represent and store information. Form-filling systems are organised around a list of slots with specifications for the kind of information that can fill in the slots. At its simplest form a dialogue is conducted by the system asking a question in order to elicit the information for each slot and the user providing the information. Normally the system has the dialogue initiative. However, there are certain possibilities for the user to take initiative, some of them defined in the form of sub-dialogues. Dialogue context consists of the collection of the values of the slots already filled in as well as the information that still has to be elicited by the user. The latter is based on the slots that are still empty. This makes the engine easy to reuse for other domains by changing the definitions of the slots. The dialogue specifications need not change. On the other hand, no dialogue history is maintained in form-filling and this design can only allow a rigid dialogue context. An example of such a system is the Philips train information dialogue system, which is an automatic inquiry system [Aust and Oerder et al, 1995]. AutoTutor also uses a kind of form-filling dialogue management [Graesser *et al.*, 1999].

The information-state approach uses a central data structure, the information state (IS) that stores information on the history and the current state of the dialogue. IS implements the view that sees dialogue as the information that each participant has at every point in the progress. All the information that is related to the way the transitions should be handled constitutes static information. It represents the desired behaviour in a dialogue and remains constant during the dialogue. Everything that changes, or becomes updated, with the utterances is the dynamic part of the information. The way the dynamic part is updated is determined by the static information. Update rules are used for updating the representation of the information state. They include details of if and how each field in the information state should be informed by the current utterance. Updates are composed of conditions and effects. The IS is the communication platform for all other modules in the dialogue system. It can consist of different fields containing different kinds of information about the IS assumed. Divergent approaches to dialogue modelling can be implemented by defining different fields. The IS structure makes it possible to handle complex domains by defining fields to accommodate the information coming from multiple modules and integrating it in this structure. The update rules can also be complex and take care of multidimensional dialogue phenomena. The IS approach is implemented in the TrindiKit system [Larsson *et al.*, 1999; Bohlin *et al.*, 1999; Larsson and Cooper, 2000].

New technologies like the information state in dialogue management open the way to implementing complex dialogue genres like tutorial dialogues and exploring the different components of what makes it efficient. A research area that remains open is the function and automation of tutorial feedback in general and of hinting in particular in NL tutorial dialogues.

As we saw, existing approaches to tutorial feedback either do not distinguish the dialogue model from the teaching model, or they not do distinguish between tutoring and dialogue functions of hints. The cognitive functions and the dialogue move realisation of hints are also not distinguished, although there is research pointing to the necessity of such separation (e.g. [DiPaolo *et al.*, 2004]). This makes hinting non-flexible, both with regard to its content and with regard to how this content is realised in different discourse and tutoring contexts. Such non-flexible hinting hinders the naturalness of the tutorial dialogue [Tsovaltzi and Fiedler, 2005] and consequently its effectiveness. Additionally, this approach makes systems using it potentially hard to maintain or reuse as a whole.

Moreover, such approaches are limited in capturing the various underlying functions of hints explicitly and relating them to the domain knowledge and tutoring context dynamically. The state-of-the-art in this direction, which has brought ITSs a long way, is to represent the taught knowledge or the cognitive skills required in solving a task explicitly, and then use a pre-defined sequence of hints for each piece of knowledge or skill, and associate text templates with that hint sequence. Variables in the pre-written text templates can be filled in automatically for the current problem (e.g. [Anderson *et al.*, 1995; VanLehn, 2006]). Commonly, the pre-defined sequences start with an abstract hint that prompts the student for the answer, continue with a hint that provides some knowledge to help the student find this answer, and conclude by giving the answer away. It is now becoming clearer that more research is needed in the area of hinting mechanisms that provide a full automation of the content of hints, of the choice of appropriate hint, and on the on-the-fly realisation of hints, all based on the tutoring context [du Boulay and Luckin, 2001; VanLehn, 2006].

Hint production could be a bit more adaptive than the state-of-the-art hint sequences if it took the overall level of students and their motivational state into account in order to at least skip hints in the pre-defined sequence. A significantly more adaptive approach would first of all require more kinds of hints, and would present them to students when appropriate based on their current cognitive needs for the focal point in the task and their overall learner profile. It would also ideally consider various feedback strategies to handle such hints. Feedback strategies should include other pedagogical feedback to serve functions apart from those performed by hints. Strategies can implement pedagogical principles and accommodate different aspects of learning theories that seem to be competing, but actually only apply to different learners and situations. Without the adequate amount of adaptivity, developers are forced to make choices on which of such aspects to implement, where a choice is not really necessary or appropriate. For instance, whether to provide feedback on request or unsolicited feedback is not a matter of an absolute context-independent decision, but depends on the situation. Students who ask for too much help should be directed into trying harder on their own. To take the other extreme, a student who is not progressing in the task and does not request feedback should be given unsolicited feedback. There can also be various other student behaviours in between. With the adaptive approach we sketched, this kind of decision is possible on the fly. In general, adaptivity of feedback would be possible in the choice of strategy, in the choice of pedagogical feedback or hint inside the strategy, and in the order these are presented.

From the existing systems, some employ more and some less pedagogical principles from learning theories in defining hints. Nonetheless, when such principles are used they remain general guidelines for choosing feedback, for example to provide immediate feedback or not. They are not captured in any structured way in the choice and composition of the feedback itself. This would involve breaking down feedback to its composite functions across processes, that is cognitive, metacognitive and affective processes. Such functions should be defined at a fine-grained level to capture all the constituent functions of one unified tutorial feedback.

Moreover, the huge cost of building NL dialogue ITSs makes the need to build reusable systems for different domains imperative. A first major step in this direction would be accomplished by separating dialogue and tutoring capabilities. An ITS would then be reusable if it implemented general tutorial dialogue techniques that are common among all domains in this genre. Tutorial management can be built on top of this domain-independent platform. Tutorial management itself should be conceptualised and implemented in a way that it may be reused for different domains. This means that the pedagogical principles captured in tutoring strategies should be content free. That is, the content of feedback must be distinguished from the tutoring function of the feedback. Consequently, the same tutoring strategies can be used for different domains. Under this scenario, feedback content is the only level where additional implementation effort is needed to build an ITS for a new domain. If there are also guidelines on how to represent such knowledge so that the content of the feedback may be chosen automatically as well and so that it may be integrated with the pedagogical knowledge and the NL dialogue knowledge, then this would take ITSs yet another step further towards adaptivity.

This thesis undertakes research on automating tutorial dialogue feedback that contributes to adaptivity in the above directions. It concentrates on automating feedback as a means for implementing adaptive teaching strategies. It looks at the general possibility of combining flexible dialogue management of the kind we reviewed and explores the integration of automatic feedback in a NL dialogue tutorial system. It argues that NL dialogue is most suitable for implementing the required adaptivity in one unified system output with functions on multiple levels and dimensions. One level represents the distinction between dialogue and tutoring functions. Each of these two comes with its multiple dimensions. In dialogue, multiple dimensions represent dialogue functions. Respectively, in tutoring, multiple kinds of feedback represent cognitive, motivational and other pedagogical feedback. The function of hints is also a special case, which is itself defined across multiple dimensions.

In terms of pedagogical principles and teaching strategies, the thesis investigates the automation of a Socratic teaching strategy. It defines a domain-independent teaching model based on prominent learning theories with schema theory at the foreground. It specifies domain-independent pedagogical feedback for realising the model (e.g. motivational feedback) and scrutinises hints with its many dimensions as the main and most complicated kind of feedback in this model. Multiple cognitive functions are defined, which are captured in different hint dimensions. They are domain-independent apart from the one where domain knowledge comes into play. However, an approach to structuring the tutoring domains and defining instructional points so as to represent schema theory is suggested, with emphasis on problem solving domains. This structure allows using the represented knowledge to deliver domain-independent feedback that promotes schema acquisition. The choice of feedback is handled by substrategies that realise our Socratic teaching strategy. Eight out of the total ten substrategies are domain-independent. The other two substrategies that handle domain-knowledge may be used for other domains to the extent that they can be represented by the abstract definitions of instructional points that we suggest.

Overall, the thesis suggests an approach to automatically producing student, situation, and discourse adaptive feedback and hints. An example of such adaptivity that our approach can handle is to decide on-the-fly about providing solicited or unsolicited help. A further decision that it handles is what kind of form this help should take. Moreover, the fine-grained analysis and representation of the components of tutorial feedback enables yet another form of adaptivity, namely dialogue and discourse context adaptivity. As the production and content of hints are adaptive, there is no need for pre-formulated NL dialogue feedback. This opens up the possibility of generating automatic NL realisations and thereby also of making the NL realisations of the feedback dialogue and discourse sensitive. Our analysis of tutorial dialogue functions constitutes a contribution to this end.

Finally, the thesis presents the tutorial manager `Menon`, which implements the suggested approach to automatic production of feedback for problem solving. It further produces output that can be used for the automatic dialogue and NL generation of feedback.

### 1.3.2 General Approach to NL Tutorial Dialogue

Our approach is oriented towards integrating adaptive feedback in NL tutorial dialogue [Tsovaltzi and Karagjosova, 2004] and allowing for dynamic NL realisation of tutorial feedback. In this context, and in the framework of the DIALOG project [Benzmüller *et al.*, 2003a], a prototype NL tutorial dialogue system was built [Buckley and Benzmüller, 2005; 2006]. This prototype proposes a variant of the typical ITS architecture, and focuses on NL dialogue management and flexibility (Figure 1.1). The system depended on an existing specialised domain reasoner, the proof system OMEGA [Siekmann *et al.*, 2002] for the task management in set theory. This system design enables reasoning about the student's action for arbitrary problems in the domain and bears the potential of elaborate adaptive system feedback without the need to pre-compile it for each problem.

More specifically, the dialogue manager in Figure 1.1 is responsible for managing the dialogue and for the communication among the different modules. It is based on the information-state approach that can handle the amount of complexity of tutorial dialogues and our multidimensional approach to tutoring. Every module connected to the dialogue manager provides information about its own state, which is also written in the IS. Update rules are responsible for changing the IS based on incoming utterances by the user or changes in the state of the other modules in the system. The discourse theory is also implemented in the update rules. All communication between the modules is done through the dialogue manager and the IS.

Before we explain the architecture in Figure 1.1 in more detail, here is a snapshot of the kind of dialogue that can be handled by such an architecture. In brackets, we provide the dialogue moves produced for the tutor turns and the evaluation of any domain contributions for the student turns[2].

[2]Throughout the thesis, we use the following font conventions:
names of hint dimensions → SCRIPT-SIZE SMALL CAPITALS
names of dimension classes → `script-size mono-space`
names of hints → `mono-space`
names of dialogue moves → *slanted*
names of strategies and substrategies → *`slanted mono-space`*
names of functions → SMALL CAPITALS
Hinting Session Status and its subfields → sans-serif

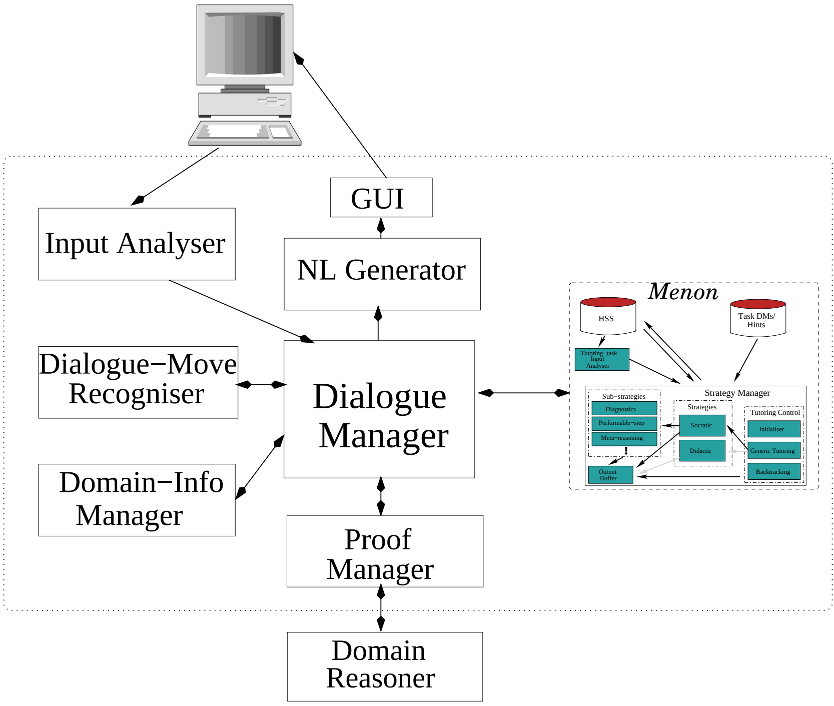


Figure 1.1: The architecture of the prototype NL tutorial dialogue system of the Dialog project

**T0:** (`initiate-dialogue`) Hello.

**S0:** Hello.

**T1:** (`initiate-task, prompt`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset of K(B), then B is a subset of K(A)!

**S1:** (`partial-answer(ia)`) I have to identify what's given and what I have to prove.

**T2:** (`encourage`) Great! (`signal-pa`) You're on the right track. (`initiate-subtask-proof-step-meta-reas`) We're taking it from the start. So, go ahead and (`elicit-prem-conc`) find what is assumed and what you have to prove.

---

names of relations → *slanted sans-serif*
names of instructional points → sans-serif and capital first letter
We also use *italics* where a term is defined for the first time or is extensively discussed.

**S2:** (`correct`) I have to prove that $B \subseteq K(A)$, and $A \subseteq K(B)$ is assumed.

**T3:** (`signal-accept`) Correct! (`elicit-specific-method`) Now, how can you manipulate the expression to prove what you want?

...

The first two turns **T0** and **S0** do not have any tutoring content, so they require NL analysis and dialogue management, but any modules that deal with domain knowledge or the tutorial manager (`Menon`) do not have to be involved in dealing with them. **T1** sets the task, which must be chosen by the domain reasoner. `Menon` produces the dialogue moves and specifications for them. The proof task must be instantiated with the chosen task for the NL realisation of the dialogue move *initiate-task*, as this is its domain-content specification defined by `Menon`. In **S1**, the student makes a first attempt at solving the task. This fact must be recognised by the system, which has to identify the dialogue move performed as a *domain-contribution*. *Domain-contributions* must be analysed for their domain content in order to assess their correctness. They must also be analysed for any further relevant aspects for tutoring, for instance, if there's reason to believe that the student is demotivated. Based on this analysis, the tutorial manager can provide appropriate pedagogical feedback. The production of the dialogue moves in turn **T1** by the tutorial manager are the result of this analysis of **S1**. The NL realisation for the moves in this turn can be done by only taking into account the dialogue and the discourse context. For example, the move `elicit-prem-conc`, which is a hint, may be also realised as "Can you tell me now what is assumed and what you have to prove?", or "I would like to know what is assumed and what has to be proved", and so on. These realisations refer to the same domain-content as the one in the example – the premise and the conclusion – but would sound strange in the specific dialogue context. The first one because it sounds as if the student has not already suggested what the hint is asking her to do, and the second because it sounds way too demanding in the context. Both may put the student off. **S2** is similar to **S1**, but it additionally has to be assessed for whether it identifies the premise and the conclusion correctly in the particular task. The tutorial manager needs to know this before it can consider them known and can choose the next domain knowledge to address. Since the premise and the conclusion are correct, the tutorial manager decides in **T3** to move on and draw the student's attention on the method to use for tackling the step[3].

Let us now turn to the architecture of the DIALOG demonstrator. To be able

[3]Note that the student's answer does not name exactly the premise and the conclusion of the deduction rule *Implication Introduction*, whose premise is $B \subseteq K(A)$ under hypothesis $A \subseteq K(B)$ and its is conclusion $A \subseteq K(B) \Rightarrow B \subseteq K(A)$ (cf. Chapter 3). So, the answer is underspecified from a logical point of view. However, for the purposes of the proof this answer is sufficient and the proof manager should categorise it as correct. The proof manager should also provide the same instantiation for the instructional point premise-conclusion for the NL realisation of the relevant hint that gives it away. For more on this phenomenon that is investigated by the OMEGA group and is considered assertion level proving, we refer the reader to [Vo *et al.*, 2003]

to handle this kind of dialogue, the IS in the Dialog prototype is informed by the following modules:

**GUI (graphical user interface)** : The GUI accepts typed input from the student and presents the system's feedback.

**Input Analyser:** The input analyser is responsible for the NL analysis and it provides an underspecified representation of the mathematical content in the student's input.

**Dialogue Move Recogniser:** The dialogue move recogniser is responsible for assigning dialogue moves to chunks of input based on the dialogue state.

**Proof Manager:** The proof manager monitors and maintains the proof task, and provides information relevant to the evaluation of the student input.

**Domain Information Manager:** The domain information manager determines the content of the mathematical information for the proof step at hand. It is responsible for identifying this information in the student's attempt and for the instantiation of the domain-dependent hint specifications, formalised in a domain ontology (cf. Chapter 5). It also represents declarative knowledge, ie. definitions, theorems, lemmata etc.

**Tutorial Manager (`Menon`):** The tutorial manager is responsible for the pedagogical feedback. It manages all task dialogue moves and chooses the kind of hint or other task move to be produced as general pedagogical feedback. It provides the domain-dependent specifications for the NL realisation of the dialogue moves.

**NL Generator:** The NL generator is responsible for the sentence realisation. It uses pedagogical output and takes into account dialogue and discourse considerations to generate a NL realisation of the dialogue moves that make up the system's response. This is the overall system output, which is shown in the GUI.

The information flow in the system, which is regulated by the dialogue manager, is as follows. The student inputs an utterance in natural language in the GUI, which can be seen in Figure 1.2[4]. The input analyser specifies the linguistic meaning of the input and builds an underspecified representation of its mathematical content. The proof manager communicates with the domain reasoner Omega, and attempts to restore a full representation of the student's attempt from the underspecified one built from the input analyser. It, hence, determines the proof step attempted, and evaluates the student attempt for relevance, granularity, and completeness. The dialogue move recogniser assigns dialogue moves to the student input in the domain, based on its linguistic meaning.

The dialogue manager makes decisions on the functions of the output dialogue moves apart from the task dimension. The domain information manager

[4] This interface was originally developed by [Fiedler and Gabsdil, 2002]

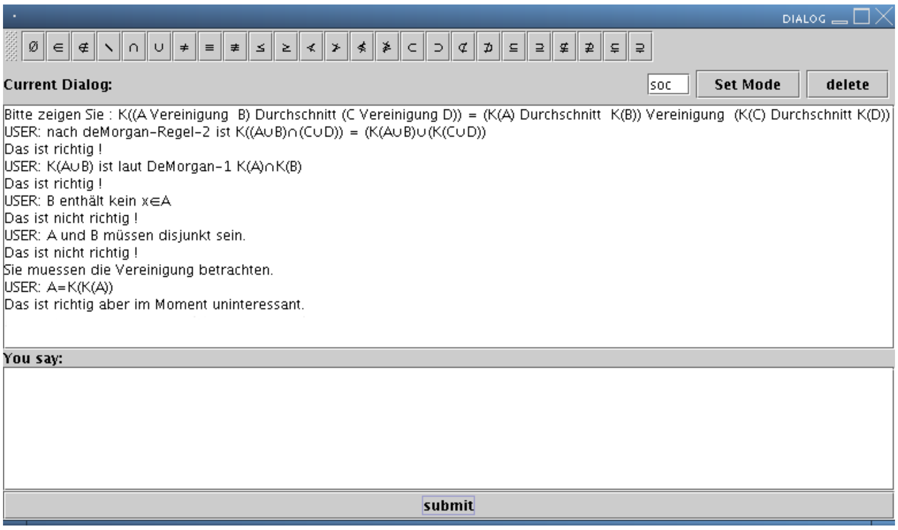


Figure 1.2: The architecture of the prototype NL tutorial dialogue system of the DIALOG project

compares the student's attempt to the complete step and assigns finer-grained evaluation, e.g. whether the step is a near-miss, or just wrong, or which of the expected domain information is present in the student's attempt. The tutorial manager represents the state of the tutoring task, and implements a tutorial strategy that decides which dialogue moves should be produced by the system next at the tutoring level. It is responsible for producing adaptive pedagogical feedback. The domain information manager instantiates any domain-dependent specifications that are included in the tutorial manager's output. Finally, the NL generator realises the system output by putting together all the functions of the dialogue moves that have been determined by the dialogue manager and the tutorial manager along with the domain-dependent specifications, and further discourse considerations, such as discourse markers, to generate an NL realisation. This is output to the GUI and the system is ready to receive the next input. At the time of the development of the prototype system, the dialogue move recogniser, and the domain information manager did not produce their output automatically, but this was hard-coded for specific proofs. Relevant research for the development these modules since then. For the dialogue move recogniser see [Tsovaltzi and Karagjosova, 2004], Chapter 4 and Appendix C. For the domain information manager, see Chapter 3 and [Autexier *et al.*, 2004; Autexier and Fiedler, 2006].

The tutorial manager used in this prototype system was conceptualised and designed as part of this thesis and was the first stage in the development of the approach to adaptive feedback that we investigate here. This first naive implementation was done by the Omega group. In this thesis, we present the enhanced and full version of the tutorial manager `Menon`, which involved the extention of our original concept and design, as well as the reimplementation of the first naive version.

### 1.3.3 Our Approach to Automatic Feedback for NL Tutorial Dialogue

In the context described in Section 1.3.2, the present research takes into account the different constituents of tutorial dialogue that make it efficient, allows for the possibility to integrate them, and concentrates on the pedagogical aspect and the cognitive functions of feedback. In particular, it scrutinises the multiple functions of hints within tutorial dialogues. Our aim is to dynamically produce general tutorial feedback and hints for arbitrary proof tasks, which fit the needs of the student with regard to the current proof. Dialogue management can then adjust this automatic feedback to the dialogue and discourse context and a NL generator can provide the final verbalised output to the reader. The tutorial manager `Menon`, implements this approach to automating feedback.

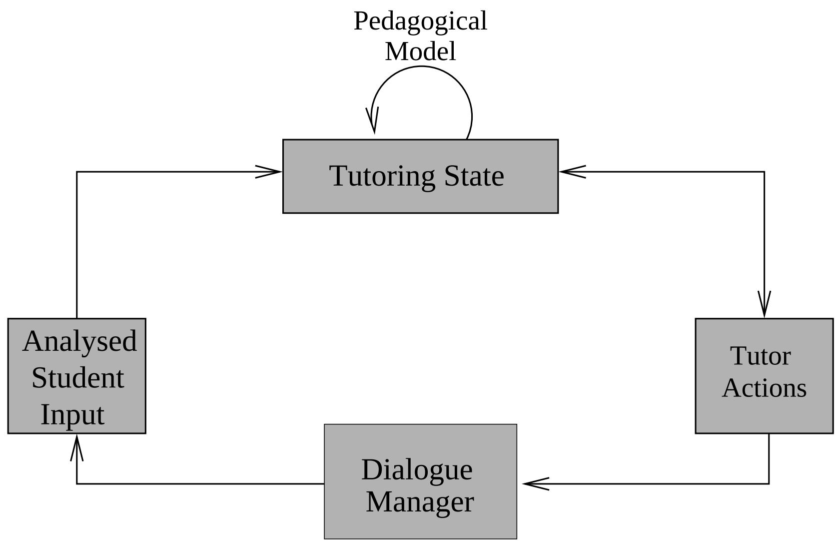


Figure 1.3: Tutorial Manager Cycle

`Menon`'s output implements a non-goal-specific instructional tutoring model, which allows the students to build their own knowledge on existing mental structures and form helpful schemata without super-imposing a particular cognitive model. To that end, *proof step matching* allows matching the student's attempt to one of the possible correct proof steps, which we call the *expected proof step*, and the subparts of this attempt to the expected subparts. The result of comparing the expected step with the student's attempt is an evaluation of the student's input, which depends on domain knowledge and constitutes input to `Menon`. Proof step matching is similar to model tracing in the sense that more possible solution paths are available, but it is more dynamic, as the use of a theorem prover, which it presupposed, makes the explicit definition of these paths obsolete. We also use tutoring concepts, which we call instructional points, and define them in a domain ontology (cf. Chapter 3) that allows matching specific mathematical concepts, relations and terms in the proof step, to the abstractly defined instructional points. *Instructional points* are abstract representations of the knowledge required for solving a problem. The tutor points the student to the instructional points, so that the student can acquire the corresponding knowledge by looking at particular steps in the task. This aspect of instructional points is equivalent to the notion of *knowledge components* [VanLehn, 2006]. The extraction of hint categories is based on the defined instructional points. This way, `Menon` produces feedback for the proof step that the student attempts on the fly and there is no need to impose a precompiled solution. However, our hinting strategy provides help primarily when the system judges that help is needed, and not when the student asks for it. Evidence supports that the latter may be a weak tactic as students cannot generally monitor their own progress and need for help [Aleven and Koedinger, 2000b].

We borrow the theoretical construct schema from cognitive psychology to explain learning to define a blueprint for hinting. Traditionally, a *schema* has been seen as a complex structure for organising knowledge. It may be used to represent objects, relations, perceptions, situations, events, and sequences of events. It comprises concepts connected by relations that represent the structure of the world [Russell and Norvig, 2003]. Most importantly for our work, according to Rumelhart and Ortony [Rumelhart and Ortony, 1977; Rumelhart, 1980], a schema is a network representing meaningful knowledge that is built actively through experience[5]. With the notion of schema in mind, we define a blueprint to use as our hint-choice strategy. This blueprint is a possible heuristic model for problem solving in our domain, as suggested by instructional theory (cf. Chapter 2). The definition of the blueprint is based on the relation between the instructional points, on pedagogical considerations, on the analysis of empirical data from two different domains [Moore, 2000; Benzmüller *et al.*, 2003b; Wolska *et al.*, 2004], and on the structure of our domain (cf. Chapter 3). We incorporate motivation in our teaching model both in the form of explicit encouragement captured by a dialogue move, as well as in the form of informed tutoring choices, which aim at promoting attention, relevance, confidence and satisfaction (cf. Section 2.2.4). Motivation is also promoted through the personalisation of the learning process [Ross and Fulton, 1994].

Reduction of cognitive load is another goal of our teaching strategy. Cognitive load theory emphasises the importance of providing right amounts and the appropriate level of help to alleviate extraneous load [Sweller, 1989]. We

[5]We expand on the notion of schema in Chapter 2

define the hint taxonomy with this aim in mind (cf. Chapter 4). Therefore, we identify small meaningful domain-knowledge chunks, which the hint-choice strategy makes use of for personalised instruction (cf. Chapters 4 and 6). The student's personal needs are captured in a model – the *Hinting Session Status* – that uses information of the tutoring session to operationalise these needs (cf. Chapter 5). Hence, our goal is that the only source of cognitive load for the students is the organisation of the knowledge provided by the instructions, which is relevant to the schema acquisition [Paas *et al.*, 2004]. Our teaching strategy also encourages forward-looking proving as long as the students can manage it, but assists students when they can only apply a means-ends technique, to reduce the unnecessary cognitive load that this imposes [Chi *et al.*, 1982; Sweller, 1989; Koedinger and Anderson, 1990]. Finally, the strategy fosters implicit learning as a mechanism for recognising family resemblance [Mathews *et al.*, 1989], which enables the re-application of learned schemata to other problems and domains. We lay out the derivation and the details of the tutorial model that `Menon` implements in Chapter 2.

At an abstract level, `Menon` works following the cycle in Figure 1.3. The tutoring state is updated every time there is a new student action. The information passed on to it by the dialogue manager is the analysed student input, which includes an evaluation of the student input (e.g. wrong, irrelevant, near-miss) and which of the expected domain information is present in it (e.g. if the correct rule of inference is used). Based on the new tutoring state, and the pedagogical model represented in its teaching strategy, the tutorial manager chooses the next tutor action. The choice involves defining which tutoring-task functions the output should have, and which domain information should be addressed next. The tutoring state is internally updated to represent information relevant to the previous tutor action (e.g. which task dialogue moves were output, which substrategy was used, which information was requested or given away). In the next cycle this information together with the new analysed student input will define the new tutoring state. The tutorial manager sends its output to the dialogue manager and awaits the next analysed input from it.

The Socratic strategy that realises the tutorial model has different components that allows divergent behaviours based on the current tutoring situation. We now describe the basics of this behaviour, which we analyse in Chapter 6. The most general tutoring behaviour accepts complete steps by the student and calls a recapitulation substrategy to recapitulate a proof when it is completed. To treat any other answers by the student, the strategies component is called. Once the hinting has started, any answer that covers a previous active hint counts as correct (cf. Chapter 4). The tutoring feedback is divided into (i) pedagogically-motivated generic output, (ii) production of appropriate hints based on the formalisation in Chapter 4. Pedagogically-motivated generic output comprises:

1. encouraging students if their performance is not optimal
2. signalling the evaluation of the students' *domain-contribution*

3. signalling the closing and initiation of substrategies when they occur and

4. prompting students to perform an action

The production of hints is managed as follows. If the student gives a non-correct answer and the tutor could not immediately diagnose the problem, a *diagnostics* subdialogue takes place in order to pick a substrategy for dealing with it. When the student input bears interesting non-content characteristics, such as a missing element from a list, an appropriate pragmatic hint is given. These hints give little information away, but can help the student nonetheless. If such a hint works, the student is motivated. In any other case a conceptual hint is chosen, which addresses the more fine-grained proof level of the proof step – the different instructional points – in order to lead the student to the complete and accurate step. To summarise the principles of such hints, without diving yet into how we operationalise the student's needs, here's a sketch of them.

1. Conceptual hints that address the performable step are given for as long as students seem to follow them and their performance is good to average. Such hints provide a moderate amount of help without explanations.

2. When students are not following anymore (many hints are given and a lot of non-correct answers follow) and they give a completely wrong answer, the proof step is given away along with an explanation.

3. If students are not following, but their answer is not completely wrong, a strategy is called, which does not give away the proof step, but leads students closely through the task, gives local answers away more easily and explains more. This strategy is stopped when the number of correct answers exceeds the number of non-correct answers.

4. Finally, if hints appear not to work at all, a hint instructs the student to read the preparation material again. After that, the interactive session and the whole hinting process starts afresh.

Parallel to this process, meta-reasoning hints, which are more related to explicit learning and explain more, are provided as a reinforcement to the performable-step hints when these do not work well alone. Moreover, we treat common misconceptions, wrong or missing declarative knowledge that is crucial to the current step, and wrong formulations and terminology use when the student level is high. If students request help, we decide whether to provide it or not based on their performance and their requests for help up to the point, and on their estimated motivation. According to the student performance, we provide regular static or interactive recapitulations of the progress in a proof step. We also allow the student to take back a turn, a proof step, or even to start with a new solution.

In Section 1.3.4 we look at an example dialogue, to give the reader a better idea of the kind of interaction that Menon supports. This example is a more extended version of the one we saw in Section 1.3.2. It concentrates on `Menon`'s approach and its automatic feedback. It is a real example of the dialogue moves `Menon` produces. Although `Menon` produces these moves as output and even the domain-dependent NL specifications for the hint moves, the natural language formulations shown in the example are not generated by `Menon`, but are made-up. We only use them to demonstrate the approach that `Menon` supports as a whole, but do not make claims as to the appropriateness of the NL realisation, which we do not investigate or implement in this thesis. A system like BUG [Callaway *et al.*, 2006], described in Section 1.2.1.5, could be used for the NL generation. BUG takes as input dialogue moves, like `Menon`'s output, and additionally rhetorical relations and the dialogue context, which are provided from a dialogue manager, to deliver verbalisations for tutorial feedback[6]. We explain particular aspects of our approach to tutorial feedback in Section 1.3.5.

### 1.3.4 An Example

One of the tasks that we used to collect data in the domain of set theory (cf. Section 1.4.2) and a valid proof that the students may perform are in Figure 1.4. The proof is in natural language as it was phrased by a human tutor. We enumerate the proof steps performed. Figure 1.4 also includes an explanation of the first step of this proof in the form of a worked example to familiarise the reader with the style of reasoning that our approach delivers. The system assumes a tutoring phase, which precedes the interactive phase that `Menon` implements, where students get familiar with such examples.

#### 1.3.4.1 Dialogue Example

In the example, student turns are marked **S** and tutor turns **T**. The dialogue begins with standard greetings and the tutor sets the task. The first tutor turn **T1** consists of setting the task and prompting the student to start. The first answer in **S1** is a partial one, since what the student states is a correct subpart, but not the complete proof step, as expected at this point. The student's answer shows an understanding of how to start handling the problem. The tutor starts in **T2** with general pedagogical feedback. The tutor first encourages the student and informs her of the status of her answer. *You're on the right track* also includes an encouragement, so the two dialogue moves `encourage` and `signal-pa` could be collapsed, for instance to avoid repetition in a following similar turn. The next dialogue move `initiate-subtask-proof-step-meta-reas` also aims to make the student aware of the discourse structure so as to increase the relevance of the following content feedback and, hence, maximise coherence. The NL realisation of it may be different, e.g. *Let's take it from the start*, when the

[6] For research work in linguistics on the use of natural language by human tutors to inform intelligent NL tutors and the role of politeness in tutorial dialogue we refer the reader, for example, to [Porayska-Pomsta and Pain, 2000; Mellish, 2004; Porayska-Pomsta and Mellish, 2004; Porayska-Pomsta and Pain, 2004; Dzikovska *et al.*, 2007]. For empirical and controlled studies on the effects of personalised and polite language in the context of ITSs and e-Learning, see [Mayer *et al.*, 2006; McLaren *et al.*, 2006; 2007]. Also see Section 2.2.4.

***The task tutored:*** If $A \subseteq K(B)$, then $B \subseteq K(A)$
***The steps of the proof:***

1. First we assume the validity of $A \subseteq K(B)$, as this is the assumption. We will show that $B \subseteq K(A)$ follows.
2. Then we take an arbitrary element $x \in B$ and show, that it has to be in $K(A)$ as well.
3. Let $x$ be in $B$.
4. Then $x$ is not in $K(B)$ and therefore according to the assumption also not in $A$.
5. But if $x$ is not in $A$, it is in $K(A)$, which concludes the proof as $x$ was arbitrary.

***A worked example of our approach for Proof Step 1:***
We start by identifying what we have to assume and prove. Here we have to assume that A is a subset of K(B) and prove that B is a subset of K(A). Then we concentrate on the *if-then* relation which is central to our mathematical expression and therefore helps us prove what we want. Now we can consider what we know about the *if-then* relation which can help us further with the step. This is that the *if-then* relation tells us what we need to prove. Taking that into account, the rule we need to use to get rid of the *if-then* relation is that *if x then y, assume x and prove y.* Finally, we substitute the *if-then* with this rule, which means that we assume that $A \subseteq K(B)$ and we will prove that $B \subseteq K(A)$.

Figure 1.4: An Example of a proof task

tutor wants to indicate that something new is starting, which is not initiated by the student. Here we use the alternative *We're taking it from the start*, as the student has already used some relevant domain information that is handled by the subtask `proof-step-meta-reas`, namely the instructional point Starting Point, which captures how one should start handling a problem. Therefore, the subtask initiation is not solely the initiative of the tutor. In indicating this, the tutor would be clarifying the structure of tutoring in the hope to help the student to follow the steps taken. We refer to domain information that we use for instruction, like the Starting Point, as instructional points (cf. Section 1.3.5). `Menon` decides that this dialogue move has to be produced and the NL generator may decide on the appropriate realisation. Just to point out the importance of this kind of adaptivity at the sentence realisation level, *Let's take it from the start* would actually wrongly indicate that it is purely the tutor's initiative to start gradually by identifying the premise and the conclusion. This is however wrong, and it may give students the impression that what they say is not being considered, which is suboptimal for their motivation. It may also confuse students as to what they are expected to do if they thought indeed that they were taking it from the start, and the tutor still prompts them to do that. Finally, regarding the hinting part of the feedback in **T2**, a conceptual hint is given, as there are no pragmatic characteristics in the student's answer that can be used for tutoring. Since the student has not provided the premise and the conclusion, which we define as instructional points, the next hint `elicit-premise-conclusion` addresses them. It is a meta-reasoning hint of a class of hints that addresses the proof step as a whole and hence also the reasoning of how to start attacking the step. Therefore, they are produced at the beginning. The premise and the conclusion are elicited because the session has just started and the student performance is good, so we want to give the student the possibility to find the answers alone, in the typical Socratic way. The hint category, which is a sub-category of the dialogue move hints, is the specification provided by `Menon`. Based on this the NL generator can formulate an appropriate sentence to ask the student for the premise and the conclusion.

**T0:** (`initiate-dialogue`) Hello.

**S0:** Hello.

**T1:** (`initiate-task, prompt`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset of K(B), then B is a subset of K(A).

**S1:** (`partial-answer(ia)`) I have to identify what's given and what I have to prove.

**T2:** (`encourage`) Great! (`signal-pa`) You're on the right track. (`initiate-subtask-proof-step-meta-reas`) We're taking it from the start. So, go ahead and (`elicit-prem-conc`) find what is assumed and what you have to prove.

**S2:** (`correct`) I have to prove that $B \subseteq K(A)$, and $A \subseteq K(B)$ is assumed.

**T3:** (`signal-accept`) Correct! (`elicit-specific-method`) Now, how can you manipulate the expression to prove what you want?

**S3:** (`correct`) I have to simplify what we are trying to prove.

**T4:** (`signal-accept`) Correct! (`close-subtask-proof-step-meta-reas`) OK. (`initiate-subtask-rel-con-meta-reas`) Let's see, then. (`elic-meta-reas-rel-con`) Try to find something in the expression that would help you simplify the problem.

**S4:** (`correct`) Do you mean the if-then?

**T5:** (`signal-accept`) Exactly! (`close-subtask-rel-con-meta-reas`) All right then. (`initiate-subtask-domain-object`) Now, (`elicit-sub-concept`) what do you know about the if-then relation which can help you handle the expression?

**S5:** (`correct`) Something about what I need to prove?

**T6:** (`signal-accept`) Exactly! (`close-subtask-domain-object`, `initiate-subtask-inf-rule`) So, (`elicit-inf-rule`) can you tell me now which rule you can use here?

**S6:** (`correct`) Yes, if X ⇒ Y, then let X and prove Y.

**T7:** (`encourage`) Good, (`signal-accept`) that's correct! (`close-subtask-inf-rule`) OK. (`initiate-subtask-subst`) Now, (`elicit-substitution`) try to apply this rule to the expression.

**S7:** (`correct`) Let $A \subseteq K(B)$ and prove that $B \subseteq K(A)$.

**T8:** (`signal-accept`) Correct! (`prompt-step`) Move on to the next step!

**S8:** (`correct`) Let x ∈ of B, we will show that x ∈ K(A)

**T9:** (`signal-accept`) Correct! (`prompt-step`) What's the next step?

**S9:** (`wrong`) Also let y ∈ of A

**T10:** (`signal-wrong`) That's not right, actually. (`initiate-subtask-proof-step-meta-reas`) Let's take it from the start again. First, (`elicit-prem-conc`) find what is assumed and what you have to prove.

**S10:** (`partial-answer`) x ∈ B is assumed and I have to prove that B subset K(A)

**T11:** (`encourage`) OK. (`signal-pa`) Almost right. (`give-away-prem-conc`) What is assumed is x ∈ B, and what you have to prove is x ∈ K(A)

...

The student answers the eliciting hint correctly in **S2**. The domain information manager can instantiate the premise and the conclusion for the current proof step with the content of **S2**. The content of **S2** is identified in the student's input by the input analyser. The domain information manager passes on the information to `Menon` that the premise and the conclusion are known. No encouragement is judged necessary at this point, as overt motivation would be too much. The step is not completed, but the student is doing well on the whole, so the next hint is produced in **T3**, which still elicits information, rather than giving it away. It asks the student for the instructional point Specific Method. In our domain, this instructional point captures the proof direction, i.e. forward or backward step. In this case the Specific Method is backward, which means decomposing the goal. The discourse cue "now" may be added to the NL realisation as shown here, to indicate that the task is progressing and help the student follow what is going on.

With her answer in **S3**, the student provides the Specific Method and covers the rest of the instructional points that are defined for dealing with the proof-step meta-reasoning. Again, the domain information manager would instantiate this instructional point for the current proof step with *backward*, and `Menon` is informed accordingly. The tutor signals the end of the subtask with a simple "OK", again to add structure to tutoring and spare students the cognitive effort of looking for this structure themselves. This is general pedagogical feedback. The dialogue move for this signalling is provided by Menon, as tutoring tasks are managed by it. The start of a new subtask is then indicated in **T4**. The hint produced still elicits information, but at the same time it explains the reasoning behind the step – the meta-reasoning – at this point, because the student has still not given the complete proof step and might need this sort of explanation in order not to feel lost and not be overwhelmed by the cognitive load. Still, hinting seems to be working and there is no need to give the whole answer away. The instructional point addressed is the Relevant Concept, and the hint explains its function in the process of proving. The definition of the Relevant Concept in the domain ontology is available for that. We will see the definitions of all instructional points in Chapter 3. Informally, the Relevant Concept is the concept that is present in the expression to be handled next and is necessary for the application of the Rule of Inference for the step. Menon provides the specifications for the hint, i.e. the hint category `elicit-meta-reasoning-relevant-concept` and the instructional point Specific Method. The NL realisation only mentions a description of the Specific Method: "simplify the problem".

The domain information manager informs Menon that the Relevant Concept is identified in the student's input in **S4**. As this is the ultimate goal of the current subtask, no further hints are necessary from it. The subtask is closed in **T5** and the new subtask is initiated. Both student and tutor (e.g. **T5**) refer to the implication as "if-then", for a more intuitive name than the logic term. Menon outputs the domain-dependent specification Relevant Concept for the next hint. However, the NL generator is responsible for this realisation of the implication relation, as well as for the realisation of the whole hint. Notice also the different way of signalling that the student input is correct from that in **T4**, how the phrasing for closing the task differs, and the discourse marker "now" before the hint. The student is doing very well, despite the slow progress, which is expected in the application of the Socratic strategy.

Therefore, no more meta-reasoning explanations are provided, but we switch back to addressing the performable step only. A hint that asks for the instructional point Subordinate Concept is chosen from the new subtask. Again informally, the Subordinate Concept is the concept in the expression that the student derives after the application of the Rule of Inference and is necessary for applying the rule. For this proof step, the subordinate concept is what has to be proven, namely that $B \subseteq K(A)$. The domain-dependent specification of the hint is the Relevant Concept, which has to be mentioned to elicit the Subordinate Concept. Moreover, the hint is formulated differently from the one in **T4**. Namely, it is a question rather than a suggestion. We have chosen the formulation based on what we think sounds better, as it is beyond our research topic to investigate when each formulation is appropriate. However, the specification of two hints in terms of hint category and domain information allows for the different realisation of them. The rest is the job of the dialogue manager and the NL generator in a system.

The next turns continue in the same fashion, until **S7** that completes the first proof step. In **T8** the tutor prompts the student for the following step without any hinting to allow student initiative anew. The student gives the complete next step in **S8**, but then gives a wrong answer in **S9**. This causes hinting to start again in **T10**, following the same schema-promoting hinting. The discourse markers "actually" and "first" are added for the NL realisation of the turn. The initiation of the new subtask uses the marker "Let's" this time, as opposed to turn **T2**, because as the student has not contributed to moving into this tutoring direction, namely the new subtask. The hint `starting-point` is left out this time, since the student has already performed a step, and it would be too tiring to repeat it. Instead, the premise and conclusion are elicited directly. In turn **S10**, the student's answer to the eliciting hint is only partially correct, so the tutor provides the passive hint that gives the answer to the previous eliciting hint away in order to boost the student's attempt a bit, but without giving away the whole step. The specifications for the hint are its category `give-away-premise-conclusion` and the instructional points premise and conclusion. Given that there was a previous hint asking for this information, the NL formulation of `give-away-premise-conclusion` indicates this. Alternatively, the hint might have been formulated as *We assume $x \in B$ and prove that $x \in K(A)$*.[7]

### 1.3.5 The Particular Aspects of our Approach

**Pedagogical Model and Teaching Strategy** In terms of feedback choice, we propose a full-fledged tutorial model based on learning theories and empirical data. We model a Socratic teaching style, which allows us to manipulate aspects of learning, such as promoting schema acquisition, help the student build a deeper understanding of the domain, eliminate cognitive load, and manipulate motivation levels [Wilson and Cole, 1991; Weiner, 1992; Lim and Moore, 2002], all in the context of NL dialogue interaction (cf. Chapter 2).

Let us look at a few advantages of our general approach with regard to implementing our pedagogical model, as opposed to standard feedback methods. We believe, like [Beal and Lee, 2005] that no matter how good the cognitive preparation of a hint is, student cannot make any use of it if they are not motivated to resolve the task and learn. Consequently, we view motivation theory as complementary to other learning theories, but inseparable from any other cognitive goal. `Menon` integrates elements of motivation theory into a unified tutoring model and binds motivation considerations with schema theory. As a whole, our approach contributes to motivation in two ways. First, dynamic feedback production may increase confidence, as students are allowed to build their own input. Second, dynamic realisation of tutorial feedback may contribute to maintaining attention, as the change in NL realisation of feedback can make the hints of the same category less boring when they have to be produced many times [Keller, 1987]. `Menon`'s feedback is produced dynamically and promotes active participation of the student, while at the same time tries to

[7] The same example dialogue can be found in Chapter 6 Section 6.6 with more elaborate comments on the operationalisation of the choices made by the strategy implemented in `Menon`.

help the students with solving the task by step-wise hints. This approach aims at increasing confidence, relevance and satisfaction [Keller, 1987; Weiner, 1992; de Vicente and Pain, 1998]. Moreover, `Menon`'s output in the form of dialogue moves that can be realised by an NL generator based on the dialogue context intends to contribute to maintaining the student's attention. Finally, `Menon` models dialogue moves that explicitly motivate the student when necessary.

Additionally, we aim at the promotion of schemata, which we represent through abstractly defined but meaningful knowledge chunks: the instructional points. Since reduction of cognitive load is a prerequisite for the acquisition of schemata [Sweller, 1989], the definition of the hint taxonomy based on instructional points takes that into account. The Socratic strategy, in turn, makes use of the instructional points to make hint choices and to provide help in right amounts and at the appropriate level in order to minimise extraneous load. This way, we try to restrict the source of cognitive load to the assimilation of the instructions that fosters the acquisition of schemata.

In general, the fully adaptive teaching strategy can make the feedback unpredictable to students to a large extent, because it implements a variety of feedback methods, and a mixture of hints and general pedagogical feedback. For example, when the student requests help, the strategy chooses between giving the answer away, encouraging the student to try harder, or requesting that the student prepares better before trying again. This choice is based on striking a balance between providing necessary instruction and fostering engagement. At the same time, the multiple possibilities of feedback constitute precautions to the students "gaming" or "abusing"the system [VanLehn *et al.*, 2005; Baker, 2007], as they cannot take it for granted that answers will be provided on request.

Such aspects taken together aim at scaffolding students to a level where active learning is possible, promoting motivation for students to work on the task alone as long as they manage, and providing the right level of feedback at appropriate times to help the student in the acquisition of schemata.

**Pedagogical Knowledge Representation** We define a domain ontology to assist us in producing adaptive tutorial feedback for arbitrary proofs. Hence, the domain ontology specifies abstract variables and relations, which we call instructional points, that are pedagogically motivated and capture human-oriented reasoning for tutoring proofs and deduction in general. These were derived top-down, first, in the attempt to define the Socratic teaching strategy, and second, by looking at existing formalisations of the domain and extracting potentially useful structures for tutoring purposes. The defined variables and relations were then specified further bottom-up, according of the analysis of empirical data, which excluded some original top-down definitions and revealed the need to define others. The final variables and relations constitute instructional points for tutoring arbitrary proofs (cf. Chapter 3). In Section 1.3.4.1 we saw examples of instructional points, such as the Starting Point, the Specific Method, the Relevant Concept, etc. The use of the variables and the relations in the ontology to define instructional points is also motivated by schema theory. By using them we aim to implement a counterpart for instruction (tutoring) for the concept of *anchoring points*, which refers to the way cognition is organised and learning works (cf. Chapter 2). Once students have acquired a schema, they can apply it to solve similar problems. Moreover, the identification of instructional points is crucial for providing automatic feedback based on schema theory, as opposed to dealing with pre-authored tasks. This approach to schema-based feedback is problem independent and can be used to tutor arbitrary proofs.

**Student Modelling** Rather than modelling the student, which is an AI complete problem, we prefer to think of our approach as modelling the tutoring session. In doing this, we avoid the pitfall of making claims about the student's actual frame of mind or affective state. We rather adapt feedback to different aspects of the tutoring session and consequently to the student progress as it is reflected in it. These aspects include:

1. information on how the proof task is evolving in the tutoring session and in the current step, e.g. if the proof step or the proof has been completed

2. information on how the tutoring task is evolving in the tutoring session and in the current step, e.g. which were the previous hints provided, or substrategies applied

3. knowledge that helps to assess the demonstrated level of understanding, e.g. which instructional points are present in the student input

4. knowledge that helps to assess the demonstrated level of motivation and the imposed cognitive load, e.g. how may hints have been given without much success in helping the student understand, and if the student resigns from the task directly or indirectly.

Part of what is represented in the analysis of the student input are the instructional points. Instructional points assist this analysis in so far as they direct the search for required knowledge. They constitute the information that should be present in the student input for an answer to be complete. If this information is missing, it should be addressed during tutoring. *Proof step matching* is used for that purpose. This is a technique for matching the student's attempt to one of the possible correct proof steps, which we call the *expected proof step*, and the subparts of the attempt to the expected subparts. The instructional points are matched to specific mathematical concepts, relations and terms in the proof step. The result of comparing the expected step with the student's attempt is an evaluation of the student's input, which depends on domain knowledge and constitutes input to `Menon`. The use of instructional points enables some shallow student modelling (cf. Chapters 3 and 5), which is characteristic of expert tutors. We use it in lack of evidence that more elaborate methods are more effective [Wiemer-Hastings, 2004]. Our approach also follows the behaviour of human tutors who take the student's input into consideration, but do not scrutinise the student's knowledge before offering help [Graesser *et al.*, 2001; Narciss, 2003].

**Dialogue Move and Hint Taxonomies** To enable adaptation for the various aspects of tutoring feedback, we have built a dialogue-move taxonomy for tutorial dialogues, which includes a taxonomy of hints (cf. Chapter 4). In this respect, we abide by the theoretical definition of [Narciss, 2003] that sees informative feedback as "a multidimensional instructional measure" (p. 22). We consider most of the feedback functions identified in the analysis of this research (cf. Section 1.2.6), leaving out metacognitive functions, which are not part of our tutorial goal and therefore we do not address them in any direct way. We also dive into the cognitive function of hints and contribute additional subfunctions that deal with: (i) the amount of information provided or elicited, (ii) views on discovering an inference that the feedback points to, and (iii) the type of inference about the information addressed by the feedback. Moreover, we move one step further from this theoretical work and define our multidimension and multilevel taxonomies as a means to capture if and how these multiple dimensions relate to each other and may be combined in one feedback message. This perspective to tutorial feedback is reflected in the different dimensions of the dialogue-move taxonomy, which models the fact that one dialogue move may have various functions that can be combined. It is also reflected in the way multiple dimensions are interleaved inside it. In our approach, *hint* is considered a dialogue move in the task dimension, which we define for the proof and the tutoring tasks in the tutorial dialogue genre. The cognitive function of hints is only one of its functions as a dialogue move, i.e. the task function. Given its complexity, we define a multidimensional taxonomy of the cognitive functions of hints. Each dimension in the hint taxonomy defines a decision point for the associated hint function. Hint categories are points in the space defined by the dimensions, where the decisions for the different functions are the coordinates of the points [Tsovaltzi *et al.*, 2004a]. We propose how this can be combined to produce pedagogically meaningful hint categories. For example, in turn **T4**, in Section 1.3.4.1, the hint category `elicit-meta-reasoning-relevant-concept` captures the meta-reasoning in one dimension, and refers to the Relevant Concept instructional point, which is a choice in a different dimension. It also refers to the domain information conceptually and is an active hint. These four decisions make up the category. A use of this explicit representation that we explained in our example dialogue is that the meta-reasoning function can be applied when a change in the student performance occurs that requires it, while the other hint functions can be kept the same. We emphasise making the cognitive functions of hints explicit in order to be able to use these functions in the choice of hints. To this aim, we define situations where separate hint functions are appropriate, rather than situations where a whole pre-compiled hint is appropriate. This adds to the adaptivity of feedback, and allows us to capture their educational power. We also provide a way to formalise the hint categories and thus make the automation of multifunctional feedback possible within an ITS (cf. Chapter 3).

**Automatic Feedback Choice** The appropriate feedback is chosen automatically by help of a selection procedure, which implements the Socratic teaching strategy. Hints and other tutoring-task dialogue moves are the basic units of the teaching strategy concerned with cognitive feedback. The Socratic strategy chooses task dialogue moves, which are domain independent but strategy dependent. For example, whether to signal the end and beginning of subtasks, which we saw in Section 1.3.4.1 in the example dialogue, is a domain independent decision. However, it must be among the goals of the tutoring strategy to make students aware of the structure of the task in order to reduce the students' cognitive effort and allow them to concentrate on other aspects of the task. This fact renders it strategy dependent.

The Socratic strategy also calls substrategies. Substrategies apply to different hinting situations and make a number of decisions.

1. They decide whether to produce a hint or another task-dialogue move, e.g. an encouragement.

2. They choose which dialogue move to produce and its parameters, representing some pedagogically motivated feedback. For example, a general encouragement produced because the student seems to be struggling may be realised as *It's a bit difficult, right?*, or *You're doing well!*. Whereas an encouragement produced after the student has requested an evaluation may be realised as *It's not easy, but you'll get there!*.

3. They specify which functions the hint should have, e.g. whether the information addressed should be elicited or given away

4. The order in which feedback will be produced, e.g. in case of multiple feedback messages.

5. They decide when the feedback will be produced, e.g. misconceptions are not treated immediately, but after the system has checked their origin.

Substrategies can be called recursively and within one another. They are divided into subtasks and subdialogues based on their primer function. That is, respectively, they give feedback to the student input directly, or they initiate a subdialogue, which assists the system in choosing the appropriate task feedback. Subtasks are part of the main task, which is the overall task of tutoring the proof. They may implement a way to deal with specific situations in tutoring, like misconceptions. Subtasks may also pick one of the hints that belong to the same class of instructional points. We refer to these subtasks as *class subtasks*. They take into consideration the tutoring situation to choose the appropriate functions of the hint to produce, and hence choose a hint category. More specifically, choosing the appropriate formulation of the dialogue move *initiate-subtask-proof-step-meta-reasoning* in **T2** of our example dialogue in Section 1.3.4.1 has no bearing on domain decisions. Moreover, any domain that uses some form of rules of inference, that is deduction, will require that a rule of inference is tutored when the student has not provided it yet.

In effect, the procedure chooses hint categories through the choice of the right cognitive function in every dimension that is represented in:

1. the choice of subtasks
2. the choice of instructional point for tutoring within class subtasks
3. the choice of eliciting or giving the instructional point away also within class subtasks

In this thesis, we implement the Socratic strategy that chooses the automatic feedback and the model of the tutoring session with its different fields that operationalise it. We also define the dialogue moves that are relevant to tutoring and a domain ontology for automatically instantiating the domain-content. This can be used for recognising instructional points in the student's answer and for instantiating the instructional points that constitute domain-content hint specifications. However, we do not implement the ontology that analyses student input and the NL generation of hints are not part of our research. In Section 1.3.2 we discussed the different modules responsible for analysis and generation in the Dialog project, and we provided references for their work in this direction.

**Distinction between NL Dialogue and Tutorial Management** We use the domain ontology in the dimension that captures domain knowledge, and show how the definition of instructional points makes domain knowledge readily available for tutoring decisions and automatic generation of hints within a NL tutorial manager. Namely, instructional points inform both the choice of an appropriate hint category and the specifications of the actual hint to be produced [Fiedler and Tsovaltzi, 2005]. For example, in **T4** in our dialogue example in Section 1.3.4.1, the hint `elicit-subordinate-concept` is chosen because this instructional point was not yet used in the student input so far, although all other instructional points that precede it in the reasoning for the step have been covered. As we saw in the example, such instructional points include, among others, the Starting Point and the Relevant Concept. Therefore, at this point the analysed student input that is sent to `Menon` must represent whether the Subordinate Concept is present in the student input.

In order to make hint realisation adaptable to the dialogue and discourse context, we define each hint category based on the instructional points. Since the hint content is chosen automatically, there is no need for pre-formulated NL hints. The automatic choice of content is only one of the aspects that define a hint. However, when the content is automatically determined, the other cognitive functions of hints as well as the dialogue functions that a hint should have in a given dialogue context can also be determined automatically. The same holds for discourse decisions. We complement those with other domain information, which assists the NL realisation of the hint. For example, the specifications of the hint `give-away-inference- rule` include the name of the rule of inference and the Relevant Concept. For the rule of inference in the first step of our example proof task in Figure 1.4 the name for the rule is Introduction of Implication and the Relevant Concept is the implication. These values for the hint specifications can be instantiated by the module in the dialogue manager that represents the task with its mathematical knowledge, e.g. the domain information manager, and can be passed on to the NL generator. The NL generator can then produce a NL realisation like *You have to get rid of the "if-then" relation.* Since the ontology defines the instructional points like Rule-of-Inference and Relevant Concept, it enables the mapping of the generic descriptions of domain knowledge onto the actual objects or relations that are used in the particular context, that is, in the particular proof and the proof step under consideration. Finally, the hint category itself constitutes the specific dialogue move in the super-category *hint*, which is represented in the task dimension of the dialogue-move taxonomy and should be realised as all dialogue moves based on information defined in the NL generator for its realisation. To illustrate this, the active hint `elicit-inference-rule`, which we saw in Section 1.3.4.1, does not need domain-dependent specifications as it does not name any instructional points. The NL generator can formulate it as *Which rule can you apply here?* based only on NL dialog-move specifications, just as it may formulate the move *signal-non-understanding*, which is not a task dialogue move, as *What do you mean?*. Over 40% of the dialogue moves that `Menon` produces as output are of this kind, and almost 15% of them are hints. The rest are hints whose domain-dependent specifications are abstractly defined.

Figure 1.5 abstracts from the specifics of the prototype NL tutorial dialogue system and illustrates how the instructional points and the other hint specification are used for the automatic NL hint realisation. The domain information manager can use the definitions of the instructional points in the domain ontology to compare the representation of the incomplete proof step attempted by the student input (which is derived by the input analyser) to the complete proof step (which is identified by the proof manager) in order to track the instructional points that are present in the student input. Again for the hint `give-away-inference-rule`, the input analyser would send a representation that does not include the Rule of Inference, the proof manager would return a complete representation of the step where the Rule of Inference is the *Definition of Powerset*. The domain information manager, will look for the instructional point Rule of Inference and will recognise that the *Definition of Powerset* is not present in the student input and this information will be passed on to `Menon`. This is represented in `Menon` as part of the new tutoring status. The Socratic strategy will decide that the hint to produce is `give-away-inference-rule` and return this hint category and the domain-dependent specifications for the category discussed above. The domain information manager will instantiate Definition of Powerset for the name of the rule, and powerset for the Relevant Concept. These instantiated hint specifications and the hint category will be passed on to the NL generator, which will integrate them with additional dialogue and discourse specifications and generates the final NL hint realisation.

Note that by providing hint categories and domain-dependent specifications as `Menon` does, the final sentence realisation of hints can also become dialogue

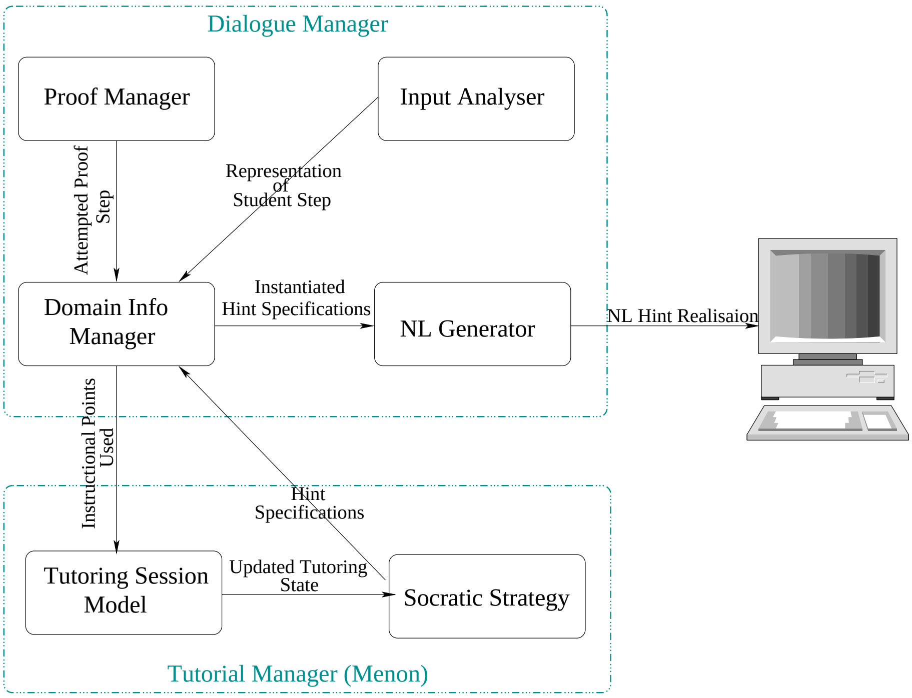


Figure 1.5: The use of the instructional points and the other hint specifications for automatic NL hint realisation

adaptive, in that any dialogue and discourse specifications can be added to these tutorial-feedback specifications and influence the actual NL realisation of the feedback. This gives tutorial feedback the best of both worlds in tutoring dialogue systems: adaptive cognitively-oriented domain-specific feedback due to the definition of hint categories as dialogue moves and the specifications of hints using instructional points, and dialogue adaptivity in tutoring, due to the possibility of taking other dialogue and discourse content into account. We saw examples of such dialogue and discourse sensitivity in Section 1.3.4.1, e.g. adding discourse markers and choosing one dialogue move formulation over another. This aspect of our approach is similar to the approach for managing tutorial dialogues in BEETLE [Zinn, 2002]. BEETLE suggests an architecture and a planning mechanism to enable such separate manipulations of the different aspects of tutorial dialogues as we envision. In this thesis, we look into the same phenomenon from the perspective of automating feedback in NL, which can be situated in an architecture like the 3-tier planning in BEETLE. Our approach is an equivalent micro-perspective of the tutorial dialogue management considerations investigated in BEETLE.

As we have argued already, separating general pedagogical feedback from tutorial feedback, and that from dialogue moves that are not related to the task, allows simultaneous processing and fast discourse manipulations that would otherwise be impossible. An example of such manipulations is back-channel feedback [Graesser *et al.*, 2001], which is far more frequently provided than other feedback, a lot of the time before students have completed their input. To this end `Menon` provides conversational cues to structure the discourse in the form of general pedagogical feedback. As we saw in our example dialogue, such cues include signals of opening and closing tasks, subtasks and subdialogues, e.g. *OK* in places like "OK. Let's see then." Moreover, the architecture suggested also supports the implementation of back-channel feedback to pump the student input on the fly, by dealing with them before the tutorial manager has provided its feedback, as such phenomena can be directly handled by the dialogue manager, which manages discourse structure.

Another advantage of our approach to automating hinting is that we can promote natural argumentation [Reed and Grasso, 2004] in two fashions. First, it is captured in the dialectic of reaching a common ground towards resolving the proof task, which we implement by the Socratic teaching strategy. The two parties participating in the dialectic are the student, collaborating with the tutor to find a proof and to learn, and the system, trying to reason with students on the basis of their input but towards a valid solution. This models natural argumentation with its characteristic dialogical structure and expressive power. Second, the fact that we keep the mathematical logic as the basis of our domain ontology, but abstract from it and capture it in a human oriented way for the definition of instructional points, is also characteristic of natural argument [Tsovaltzi and Fiedler, 2003a].

**Reconfigurability** An important effect of our feedback design is that dialogue and discourse management can be independent of tutoring strategy. With respect to discourse management, we provide an analysis of pedagogical feedback in terms of dialogue moves, which constitutes the infrastructure for implementing the theory of social obligations [Traum and Allen, 1994; Matheson *et al.*, 2000]. This theory views discourse as underpinned on the deployment of dialogue moves that pose or discharge obligations defined by the social context in which the dialogue takes place. Social obligations provide a solid background for explaining tutorial dialogues in general and the characteristic function of hints at the level of discourse [Tsovaltzi, 2001; Tsovaltzi and Matheson, 2002]. We take obligations into consideration for the development of our dialogue-move taxonomy.

In terms of applying this approach to other domains, we define different hint dimensions and represent domain-knowledge separately in one of the domains. The functions of hints captured in the other three dimensions and the rest of the dialogue moves are domain-independent. In effect, most of the teaching strategy and pedagogical considerations of the tutorial manager is domain independent and can be retained for different domains. Moreover, the domain-knowledge dimension is conceptualised in such a way that hint categories abstract from the specific values of this knowledge. This conceptualisation, as well as the way we have structured our domain-knowledge dimension may be used by developers as guidelines for structuring their own domains, with the closest ones being other problem solving domains. The idea behind this structure is simple and generic. It concentrates on general concepts that are common in problem solving, like:

- what is given and what has to be found or shown, which is very standard in problem solving
- what is the rule of inference that justifies a step
- how one can reason about finding the appropriate rule and
- how one applies it to the problem at hand

These are all related to the line of reasoning in finding a problem. An example of another domain where such a general approach might fit in well is physics. VanLehn and colleagues [VanLehn *et al.*, 2005] already recognise the fact that solutions in physics can be considered to be proofs, and that they can be explained based on physics principles that are used as justifications in a solution. This is equivalent to the rules of inference that are also used as justifications in mathematics proofs. As a matter of fact, the Rule of Inference is one of our central instructional points and the definitions of Relevant Concept and Subordinate Concept capture the reasoning for finding the Rule of Inference.

In conclusion, from the systems reviewed in this chapter, our work is mostly similar to but different from two other research efforts, the ITFL-model [Narciss, 2003] and the BEETLE system [Zinn, 2002]. The ITFL-model aims to provide general guidelines to ITS developers for composing tutorial feedback based on multiple dimensions. It distinguishes between cognitive, metacognitive and motivational aspects of feedback. We, on the contrary, move away from general guidelines and provide a model of the multiple dimensions of tutorial feedback and an implementation of the multiple dimensions of hints. In doing that, we also differentiate between dialogue and cognitive feedback; a necessity of which the ITFL-model seems to be aware of, but does not capture. We investigate how dialogue and cognitive feedback relate to each other, and how they should be treated separately in ITSs. Moreover, we implement pedagogically informed feedback strategies that incorporate the orchestrated use of the dialogue move dimensions, the cognitive hint dimensions, and of motivational aspects of feedback. BEETLE suggests a 3-tier architecture and a planning mechanism to enable such separate manipulations of the different aspects of tutorial dialogues as we envision and that involve the separation between cognitive and dialogue functions of feedback. However, no such analysis of feedback is undertaken or is implemented in the system. We provide and implement this analysis of feedback. Consequently, our approach is an equivalent micro-perspective of the tutorial dialogue management considerations investigated in BEETLE and of the general guidelines for composing feedback proposed by the ITFL-model. It is also the meeting point of the two, as it combines the idea of multiple dimensions of feedback, like the ITFL-model, but explores the dialogue level as one of these dimensions, which is important in the BEETLE system.

### 1.3.6 Goals and Scientific Contributions

In summary, given the above considerations, the specific goals and scientific contributions of this thesis are:

1. To propose a method for automatically producing general pedagogical feedback and hints.

   This has three aspects:

   (a) The separation of task vs. discourse.

   We propose a multidimensional dialogue move taxonomy [Tsovaltzi and Karagjosova, 2004] specially designed for tutorial dialogues, which includes an elaborate task dimension. This multidimensional approach makes it possible to capture the tutoring task aspect of dialogue moves without ignoring their other functions that are important for dialogue adaptivity. In the case of hints, it allows treating the cognitive and dialogue functions of them separately and enables adapting both based on tutoring and dialogue/discourse considerations.

   (b) Automatic production of appropriate hint categories based on the student level and needs.

   This aim is realised in the thesis by the hint taxonomy and the hinting strategy. The hint taxonomy captures the underlying cognitive functions of hints for the production of pedagogically justified feedback and defines hint categories based on those functions. The Socratic teaching strategy chooses the appropriate category for different students and different tutoring situations.

   (c) Dynamic realisation of the produced hint categories based on dialogue and linguistic context.

   We define hint categories as dialogue moves and based on abstract variables and relations, but do not make further claims about their NL realisation. We separate out the underlying cognitive functions of hints from dialogue functions, which might be common for different cognitive functions [Collins and Stevens, 1982; 1991; Tsovaltzi and Matheson, 2002]. To this end, we include the dialogue move *hint* in the task dimension of our dialogue-move taxonomy and use the hint taxonomy that captures the cognitive function of hints to handle this complex dialogue move. This allows the investigation of both aspects in isolation, and at the same time facilitates their re-integration for the natural language generation of the feedback. The actual sentence level realisation of a hint can then be based on decisions regarding the function that better serves the tutoring goals, as well as decisions regarding the dialogue and discourse context, which take advantage of NL dialogue capabilities. Although the possibility of this second aspect is provided in this thesis, it is beyond its scope to explore this issue further, except for the task dialogue moves (cf. Chapter 4).

2. To define a pedagogical model for teaching proving in our domain.

   Although cognitive models and instructional strategies exist (e.g. [Collins and Stevens, 1982; Anderson *et al.*, 1995; Anderson, 1993]), recently researchers have also started to try to combine the two traditions [Wilson and Cole, 1991]. The thesis undertakes the investigation of state-of-the-art research in the *learning sciences*, that is mainly cognitive science, educational psychology, computer science, information sciences, neuroscience, education, and instructional design. It explores the derivation of instruction guidelines for teaching proving. An added value of defining such a strategy and implementing it for use in ITSs is the careful and controlled application of such research, as it is observed that even the most experienced tutors do not consistently use the wealth of knowledge on learning, which psychological research has given us. This is partly due to the complexity of the psychological findings, and partly due to the lack of a model that unifies them to the level that tutors need in order to apply them.

3. To derive a teaching model for our implementation.

   This requires (i) making use of the guidelines derived from state-of-the-art research in learning and from our experimental data as the basis the definition of our teaching model, (ii) turning this definition into a concrete teaching strategy, our Socratic teaching strategy, for teaching proving in set theory. This is a well defined domain, used in formal proving. Proving involves using a certain set of givens, logic rules and domain rules of inference to reach a certain conclusion. In this context, we also investigate the use of hints so as to incorporate them better in our Socratic teaching strategy.

4. To provide an implementation as a proof of concept for the proposed method of automating hints and the teaching model.

   This involves implementing the Socratic teaching strategy that automatically and adaptively chooses dialogue moves and hints that are appropriate in the tutoring context. It also specifies the domain-knowledge that must be used to realise the hints, whose instantiation should be provided by another module, a domain reasoner.

## 1.4 Methodology

A well practised methodology in ITSs is to implement tutoring strategies as human tutors apply them. Although this is a valid method, we believe like [du

Boulay and Luckin, 2001] that it should be used with caution. Modelling human tactics from observation of tutor-student interactions involves a multilayer interpretation of the principles of learning that are employed. Learning theorists run controlled experiments and based on the results build progressive models and theories on learning. Educational theorists interpret these models for specific learning goals and provide general guidelines for their applications to tutoring. Tutors interpret these guidelines and are trained to use them and apply them to concrete tutoring situations. Finally, researchers reconstruct tutoring strategies from observing human tutors.

This series of interpretations may obscure the principles of learning and result in an unorthodox use of them in the third layer, which is often further burdened by personal unsubstantiated preferences of the tutors, or in the fourth layer, as observation is not a controlled research method. Hence, building a model of tutoring based on observed tutor actions should only be done with moderation. It should be used to inform those parts of the model, which pertain to the concrete application to tutoring situations. For example, the way a tutor applies a pre-defined pedagogical goal in a specific domain, or how dialogue techniques and discourse markers are applied.

### 1.4.1 The top-down vs. bottom-up cycle

We employ an iteration of different recommended methodologies for developing our tutorial module [du Boulay and Luckin, 2001]. We combine expert tutoring and bottom-up observations of student behaviour in the domain [Benzmüller *et al.*, 2003b], with top-down perspectives. These top-down perspectives include guidelines from learning theories (e.g. cognitive theory of schema, motivation theory, implicit learning), and automating considerations (formal proving and formal proof representation for instructional points) [Tsovaltzi and Fiedler, 2003b].

The specific steps of our methodology were as follows. We developed a first hint taxonomy and a hinting strategy that draws on that taxonomy [Fiedler and Tsovaltzi, 2003a]. The hint taxonomy and the strategy were based on data collected for the domain of basic electricity and electronics from a corpus that transcribed NL dialogue interactions between a human tutor applying a Socratic teaching strategy and a student [Rosé *et al.*, 2001b; Tsovaltzi, 2001]. Domain information requirements for the taxonomy and the strategy were derived by help of a mathematical ontology that captured domain knowledge for our domain: set theory [Tsovaltzi and Fiedler, 2003b]. The hint taxonomy defined two dimensions of cognitive functions of hints; one captured domain information based on the domain ontology and the other distinguished between the active and passive function of hints. The hint taxonomy was used by a hinting algorithm that produced different hints according to an implicit student model and the Socratic teaching method [Fiedler and Tsovaltzi, 2003b]. An implementation of the Socratic hinting algorithm was provided. Moreover, to classify the student's input, we developed a categorisation scheme for student answers, which drew on the mathematical ontology.

We then collected data on our preliminary approach to hinting in an experiment where participants were asked to prove simple theorems in our domain via dialogue interaction [Benzmüller *et al.*, 2003b; Wolska *et al.*, 2004]. The analysis of the collected data gave us valuable insight into the domain and into necessary improvements to our hint-choice algorithm. We augmented our preliminary domain ontology, added more dimensions to our hint taxonomy, and enhanced our hint-choice algorithm to incorporate these dimensions. The algorithm was turned into the implementation of the now advanced teaching model.

We will now look into the first part of this methodology, which involved two cycles of data analysis before we analyse the final version of the various aspects of our approach to automating a Socratic teaching strategy, for which we dedicate a corresponding chapter in this thesis.

### 1.4.2 A Pilot Wizard-of-Oz Experiment

The experiment that helped us collect data on improving our preliminary approach to hinting was a Wizard-of-Oz experiment. In a Wizard-of-Oz experiment, the participant interacts through an interface with a human "wizard" simulating the behaviour of a system [Bernsen *et al.*, 1998]. The Wizard-of-Oz methodology is commonly used to investigate human-computer interaction in systems under development. In subsequent experiments, implemented components can be substituted for some of the tasks carried out by the wizard, while preserving the overall experimental setup.

The reason we used the Wizard-of-Oz methodology is that it allowed us to formalise the aspects of the model that we want to implement in our system and have the wizard follow it. Our formalisation consisted of the preliminary approach to tutorial feedback resulting in the Socratic hinting algorithm. It also involved loose definitions of the control and the compare conditions. This way (i) dialogue data that represents the users' behaviour in interactions following the specific model is collected and (ii) early feedback on the model is acquired.

#### 1.4.2.1 Experiment Design and Procedure

The original goals of our experiment design were (i) to facilitate the collection of useful data for the formalisations and of unbiased linguistic data and (ii) to check the learning effect of the strategies. After the preliminary formalisations of our approach to automatic tutorial feedback we were able to define the original goals of the experiment more precisely. In particular, we tested and collected qualitative data on:

1. The sufficiency and effectiveness of the formalised hint categories
2. The appropriateness of the domain ontology for the automatic production of hints and the categorisation of the student's answer
3. The drawbacks and possible improvements of the hinting algorithm
4. The applicability of the student answer categories and ways to refine it

5. The use of subdialogues
6. The dialogue behaviour of the student and the tutor in the defined tutoring context
7. The use of natural language for realising the hints
8. The use of natural language in the student's answers

We do not look into the last two points, which are not part of the research presented here.

The experiment consisted of three main phases.

**Phase 1: Before Tutoring** During the experiment we first administered a questionnaire to collect personal details and data on the participants' mathematical knowledge. We asked the participants to give definitions of some domain concepts and assess their own level. We then gave them written instructions on the experiment. These included a description of the phases of the experiment and what they would be asked to do. The participants were told that they would be evaluating a tutoring dialogue system.

All participants then read a lesson material without a time limit, but 15 minutes were suggested. It included all domain knowledge needed for the tasks and some extra material. Domain knowledge comprised an introduction to the set theory, definitions of concepts, theorems and lemmata. After having read the lesson participants did a timed pre-test for the task, that is, they attempted a proof (cf. Figure 1.6: Task 1). All tasks used in the experiment were chosen based on their difficulty for the level of the participants. We targeted participants who had some background in mathematics but had not had college-level background in the domain of set theory. Prior to the experiment, the chosen tasks were administered to a class of first year computer science students in the University of Saarland who were asked to solve them. This was meant to control for a possible ceiling effect, in case the tasks were too easy. Such indications were not observed.

1. *Pre-test task:* $K(A) \in P(K(A \cap B))$, where $K$ denotes the complement
2. *Dry-run task:* $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D)))$
3. *Tutored task:* $A \cap B \in P((A \cup C) \cap (B \cup C))$, where $P$ denotes the power set of a set
4. *Tutored task:* if $A \subseteq K(B)$ then $B \subseteq K(A)$
5. *Post-test task:* $K(A \cup B) \in P(K(A))$

Figure 1.6: Experiment Tasks

**Phase 2: Tutoring** Participants were split into three groups, one for every teaching method that made up our three experimental conditions: Socratic, didactic, and minimal feedback. Participants were assigned to different strategies at random and the order of strategies was also randomised.

Participants first got technical instructions on how to use our interface and the tutoring system, which they thought they were evaluating. The interface, DiaWoz [Fiedler and Gabsdil, 2002] was a simple Java applet with mathematical symbols where the participants typed in their input, and a supporting engine that allowed online annotations and logging. After reading the instructions, the participants did a dry-run (cf. Figure 1.6: Task 2) on an easy proof to familiarise themselves with the interface and the tutoring.

When they had completed the familiarisation proofs, all students were tutored on two more difficult proving tasks (cf. Figure 1.6: Tasks 3 and 4) presented in a randomised order. Due to time restrictions, participants could run out of time during an attempt, whereupon the session would be terminated.

The people involved in running the experiment were the experimenter and the wizards. The experimenter answered questions relevant to the experiment, for example, on the questionnaires and the interface. The experimenter was the only person the participants had contact with. There were three wizards. The tutor was a holder of a masters in mathematics with experience in tutoring and was responsible for the keyboard communication with the participants. Prior to the experiment, the tutoring methods with their pedagogical ramifications were explained to the tutor in the form of instructions of what to do in different situations. The tutor was then trained on using the Socratic tutoring method, which was by far the most complicated one.

During the interactions, the tutor first classified the participant's contribution. A standard time-out for responding to the system's utterances was used, when the participant remained totally idle. Then the wizard decided what dialogue moves to perform next and verbalised them. Depending on the tutoring strategy employed by the wizard for a given participant, the obligatory dialogue moves to be performed by the tutor in one turn included informing participants about the quality of their answer on a scale from complete-accurate to incomplete-inaccurate and its in-between values (all conditions), giving hints on how to proceed further or entering into a clarification dialogue (Socratic condition), giving away the step under consideration (didactic condition), and prompting for the next step (all conditions). The tutor was also free to perform any other dialogue moves. At the end of every session, and irrespective of the performance of the participant the tutor presented the answers to the proofs on the computer screen. The classification of the contribution and the hint category produced were annotated electronically on the spot.

The tutor had two assistants who were the developers of the formalisations and the algorithm and were siting in the same room with the tutor during the experiment. For the Socratic condition in particular, the assistants told the tutor which hint to realise each time, as these were chosen by the implemented naive hinting algorithm. The way to realise it was left totally to the tutor. Moreover, although it was the tutor's decision to produce a hint at all or initiate a subdialogue, the tutor's behaviour was restricted by the assistants in cases where they judged that the behaviour would be impossible to model. For the didactic and minimal conditions, the tutor was simply informed about how to realise the respective strategy.

Two adjacent rooms with a one-way window were used. The participants were in one room, the wizards and the experimenter in the other with the ability to see the participant. Communication between the experimenter and the participants was possible via headphones and microphones. The wizards could see the student's window as well as their own. The students could only see their own screen. Each session lasted approximately two hours. Data was collected by questionnaires, in form of notes by everybody involved, and by logging the computer interaction electronically. The sessions were also videotaped to collect think-alouds. The electronically collected data constitutes our corpus on tutorial dialogues in mathematics [Wolska *et al.*, 2004].

**Phase 3: After Tutoring** All participants were asked to attempt a final task on paper (cf. Figure 1.6: Proof 5). This was a variant of the original pre-test task in that it involved using domain concepts also included in the tutored proofs. Finally, the participants were asked to fill in a questionnaire addressing various aspects of the system, including both the interface and the tutoring and its usability. The questionnaire consisted of questions that asked the participants to rate specific attributes of the system in a scale, and open questions, which asked for descriptive answers.

#### 1.4.2.2 Experiment Results and Discussion

The Wizard-of-Oz experiment was a pilot study that aimed at getting general impressions and qualitative data in order to develop our approach and its different aspects. In total, 24 participants participated in the experiment. They were students with a humanities or science background. Their prior mathematical knowledge ranged from little to moderate. Only the data from 19 participants were appropriate for final analysis. The rest were used for trials of the experiment that helped to improve the complicated experimental set-up. The small number of participants and the large standard deviations in the learning effects did not allow any significant results. However, our pre- and post-test comparison supported the didactic method, which was the only condition that appears to have learned, as shown in Table 1.1. The minimal condition, which was the control, did poorly as expected, and showed a negative effect of the strategy on learning. It was also not surprising that the Socratic condition did not do well, as the Socratic method used was a very preliminary and incomplete one and a major purpose of the experiment was to collect data on improving it. Indeed, the difference between the didactic and the Socratic condition left room for interpretation of the results and corresponding improvements to our approach.

**Interpretation of Results and Post-Analysis** We analysed the data further to identify specific areas of improvement and combine the best aspects used

| **Condition** | didactic | Socratic | minimal |
|---|---|---|---|
| **Mean** | 0.33 | -1.63 | -0.41 |
| **Stdv** | 2.84 | 2.35 | 1.99 |

Table 1.1: Descriptive statistics of the pre- post-test comparison

across conditions for our final teaching strategy. Although the results from this analysis can only be seen as indications due to the small sample, they nevertheless gave good directions for improvements and further development. On the whole, as expected the available tutoring time was too little for learning. For the Socratic method in particular, participants had a late start because of the nature of the strategy, which tries to elicit the answers from the students rather than giving them away, and hence is more time consuming, especially until the students have become familiar with this teaching style [Lim *et al.*, 1996]. Due to time constraints, sessions had to be stopped, in most cases just as the participants had started following the hints. A side-effect of the slow start was that the didactic condition participants were tutored on a larger part of every proof. In summary, the minimal feedback group saw the whole proof in the predefined amount of time, the didactic saw a fair amount, and the Socratic did not manage to get any tutoring beyond the first two proof steps because of the late start effect. There was, in effect, an advantage for the minimal and didactic conditions and a disadvantage for the Socratic group. This advantage may influence learning as it is connected to two default principles. First, that the more cases a student experiences, the higher the possibility of schema acquisition is [Delclos and Harrington, 1991]. Second, that the more complete solutions a student sees, the higher the chances are that learning will occur [VanLehn *et al.*, 2005].

We also conjectured that the didactic participants had a higher level already before the tutoring intervention. An ANCOVA test was run to test if the didactic condition participants were better in the pre-test. The didactic and the Socratic groups were used as samples, the pre-test scores as concomitant variable and the post-test scores as dependent variable. As hypothesised, there was a significant difference in the distributions of the two samples ($a = 0.05$, $F(1, 9) = 5.81$, $p = 0.039$). This sheds some light as to why the didactic condition learned more than the Socratic condition participants, even thought they did not improve significantly. Namely, the participants had already a significantly better level to begin with. On the contrary, the preliminary formalised hinting strategy was missing an explanatory aspect, which students of a lower level require in order to learn.

Some interesting results came from the post-questionnaires that the participants filled in. This data shows that despite the results of the post-test, the `Socratic` condition participants stated that they learned more about set theory than the didactic condition participants did (median$_{soc}$= 3.5 var$_{soc}$=0.67, median$_{did}$= 3 var$_{did}$=2.97). Although this cannot be counted as evidence that they did learn more, it is an indication that these participants found our hints helpful. One might hypothesise that given more time and since the participants liked the hints and tried to learn from them, a learning effect might have occurred. However, there was an indication in the questionnaire that the didactic condition participants had more fun with the system (median$_{soc}$= 3.5 var$_{soc}$=0.67, median$_{did}$= 5 var$_{did}$=2). From this we suspected that they were more motivated during the post-test, which followed immediately after tutoring. A parameter that could indicate that in post-analysis is the amount of time spent on the post-test. An ANCOVA test with concomitant variable the time spent on the post-test and dependent variable the post-test scores for the samples of the didactic and Socratic conditions, showed significance at $a = 0.05$ with $F(1, 9) = 6.39$ and $p = 0.032$. This supports our suspicion that the participants in the didactic condition were on the whole more motivated in the post-test.

Moreover, open questions on the students' opinion of the feedback highlight more interesting points. All participants of the didactic condition complained about the system feedback, for instance for not having been given the opportunity to reach the solution themselves ("I felt confused by the feedback, because the answer was given away too quickly"), or for not having received more step-by-step hints ("[*I would have liked*] more instructions towards the solution"), or for having been given too much instruction for their level ("The system could not discriminate between a beginner and an expert"). Similar complaints were not registered by the Socratic condition participants. On the contrary, five out of six participants in the Socratic condition chose aspects of the feedback as the best attribute of it. For example "[*I liked best*] the human-like questions", "[*I liked best*] that it was helpful but straightforward and the recapitulation of the proof at the end". We believe that the issues described by the complaints of the didactic participants can be taken care of by the Socratic teaching method (cf. Chapter 2). In addition, all but one participants said that they would use the system in a mathematics course at university. The participant who would not use the system had one of the best performances among all conditions, and was taught with the didactic method. This participant also explicitly stated that feedback that would elicit answers rather than give them away would have been more appropriate.

**General Enhancements** The issues discussed above should not be considered as significant results of an evaluation. However, they did give us insights into what changes, enhancements, or additions where necessary to make our Socratic method more effective, which was the purpose of the experiment. For example, the analysis of the better aspects of the didactic condition led us to search for improvements in the way this strategy was performed, our objective being to get the best of both worlds. An additional reason for looking into the application of the didactic condition, was that the human tutor who applied all three conditions, did not actually follow our definition of our didactic method and gave feedback under this condition, which would have been more appropriate for the Socratic method. There may be two reasons for this behaviour. One is that the didactic condition was less well defined than the Socratic, which was even based on an implemented algorithm, as we were actually only trying to collect data on developing a Socratic strategy. This fact allowed the tutor the freedom to use some of her own preferred tutoring style. Another reason might be that the tutor liked the feedback in the Socratic condition for which the tutor was also trained, and could not switch it off under the didactic condition. These two reasons combined resulted in producing feedback under the didactic condition that can be incorporated in our improved Socratic strategy.

The most striking characteristic of the didactic method as we defined it was the fact that the tutor gave *meta-reasoning* explanations that involved the reasoning necessary for deriving the proof step but is not directly represented in the proof steps. However, our tutor never gave the kind of long answers and explanations, which are characteristic of the traditional didactic method. In effect, not only can such meta-reasoning reinforce the creation of schemata, but it also reduces the students' cognitive load. Moreover, meta-reasoning hints are concise rule-like observations. They are equivalent to production rules [Anderson, 1993], or what tutors normally require as self-explanations in the corresponding tradition [Collins and Stevens, 1991]. As such, they provide students not possessing a schema the way to acquire one. This probably means for the Socratic condition that among the reasons they did not learn was specifically the lack of meta-reasoning hints. All the more so, as these participants were also weaker at the beginning and may have needed exactly this kind of support to reduce their cognitive effort for learning to occur. Therefore, our major improvement to hinting was to formalise meta-reasoning, starting from defining such hint aspects and conceptualising their incorporation in the hint taxonomy (cf. Chapter 4 and 6). The short explanations that our human tutor gave in the didactic condition offered themselves for this new investigation.

The tutoring that took place in the Socratic condition allowed us to observe more closely where students need instruction. Since it is the nature of this teaching style and of our implementation of it to provide more fine-grained feedback, we could observe these needs as tutoring evolved. Although we were expecting that students would need tutoring on general proving techniques, we were surprised by the lack of students' ability to use backward steps. More specifically, students were comfortable with using forward steps, although they were often not sure which rule to apply. On the contrary, they did not even seem to be aware of the backward technique of decomposing the goal. In fact, this was the main time consuming aspect of the Socratic tutoring strategy used in the experiments, and the part on which some essential tutoring took place, despite the naive formalisations available at the time and in contrast to the didactic condition whose teaching method does not allow such detailed tutoring.

**Specific Enhancements on Formalisations: Qualitative Results** As expected, we collected valuable data on the formalisations, which was our main goal in the Wizard-of-Oz experiment. We present here some indicative cases with relation to the preparations of the experiment.

*Student Answer Evaluation:* We spotted problematic or insufficient areas in the categorisation scheme that was used. We will mention only a few. First, a better definition of parts, was necessary. We originally dealt with this issue in [Tsovaltzi and Fiedler, 2003a], and a final categorisation scheme, which includes our new definition of parts is presented in Chapter 5. To mention one change, we define our instructional points as the necessary parts of a complete answer, because the data showed that we need a way to automate how tutors orient themselves in the student's input and reasons about the feedback to provide, without necessarily trying to represent exactly what the student knows. Second, the data reveals specific needs for building and using a representation that enables matching the students' contributions to the proof to an expected contribution. In particular, handling implicit inference steps, over-answering, correct irrelevant answers, starting the proof from scratch and choosing the expected (intended) proof step were all issues that could be addressed by use of the data. These phenomena relate to the tasks of different modules in the system. The tutorial manager is directly responsible for handling some of them, and is dependent on how others are handled. In Figure 1.7[8], for example, the student says something right, but the tutor cannot match it to any proof. Therefore, the student is asked to show she is getting at, assuming that the student would not be able to continue. After a couple of turns it actually turned out that the participant did not know how to continue.

**T1:** "Bitte zeigen Sie: Wenn $A \subseteq K(B)$, dann $B \subseteq K(A)$!"
[*Please prove: If $A \subseteq K(B)$, then $B \subseteq K(A)$!*]
**S1:** "$K(B) = U \setminus B$"
**T2:** "Das ist richtig, aber wie geht es weiter?"
[*That is correct, but how do you want to continue?*]

Figure 1.7: Correct but irrelevant answer

Moreover, we refined the categories in the categorisation scheme. We identified possible answers that can be considered as near-misses (e.g. bracketing problems, use of a '$\in$" instead of "$\subset$"), how these differ from misconceptions depending on the context, and possible ways to treat them. This information also helped us model substrategies as well as subdialogues, whose use was allowed in the experiment, but were not formalised. In the example in Figure 1.8, the tutor could not make sense of the student's utterance unless she substituted $\cap$ for $\neq$. The tutor tried to elicit a self-correction. The student could not provide it, although she realised from the question that she had used the wrong symbol. In addition, this kind of data pointed us to the sub-division of the original category wrong into two cases, namely wrong, the student gives a wrong answer, and the task dialogue-move resign, the student gives up without any attempt.

*Hinting:* In terms of hinting, we included in the hinting algorithm hint

[8] All examples include a translation from German in italics, where necessary.

**S5:** "wenn $A \subseteq K(B)$, dann $A \neq B$, weil $B \neq K(B)$"
[*if* $A \subseteq K(B)$*, then* $A \neq B$*, because* $B \neq K(B)$]
**T6:** "meinen Sie wirklich $\neq$ oder etwas anderes?"
[*Do you really mean* $\neq$ *or something else?*]
**S6:** "$\not\subseteq$"

Figure 1.8: Subdialogue

categories that we had already defined before the experiment (e.g. from the BEE corpus), but had not included in the preliminary algorithm, as we did not know how they should be managed in the specific domain. An example of this is the category *refer-to-lesson*. An instance of this can be seen in Figure 1.9. The student confuses two rules and the tutor refers her to the lesson to look up the rules.

**T4:** "Meinen Sie wirklich: $P(C \cup (A \cap B)) = P(C) \cup P(A \cap B)$?"
[*Do you really mean:* $P(C \cup (A \cap B)) = P(C) \cup P(A \cap B)$*?*]
**S4:** "ich denke doch: $P(A) \cup P(B) = P(A \cup B)$"
[*yes, I think so:* $P(A) \cup P(B) = P(A \cup B)$]
**T5:** "Das ist nicht richtig! Vielleicht sollten Sie noch einmal in Ihrem Begleitmaterial nachsehen."
[*That is not correct! Maybe you should look it up in your study material.*]

Figure 1.9: Refer-to-lesson, Participant 20.

In addition, we extended the hint taxonomy by adding two more dimensions to it to capture new functions that were revealed through the data analysis. We were able to make observations on how the four dimensions interact, but were also pointed to a further investigation of learning theories in order to specify this interaction in a pedagogically and cognitively motivated way. For example, the meta-reasoning function discussed above was added after additional theoretical investigation. It was assigned the role of dealing with underlying domain principles and general strategies. The way the meta-reasoning function should be used in tutoring was also defined more precisely. In addition, a pragmatic function was distinguished from the conceptual one used in the experiment. The new function captured a way of referring to domain information by means of cues that have to do with the form of the expected answer, rather than with its content. For example, the possibility of telling the student how many parts make up the expected answer was added, rather than naming these parts. We could then define more specific hint functions in this dimension. For example, the tutor explicitly connected the current state of the proof to something mentioned before. We formalised this into the pragmatic hint function `point-backwards`.

We used the same data for enhancing the hinting algorithm. For one, the student model that was implicit in the algorithm was separated from it. The original student model gave rise to an explicit and enhanced version, which is our current modelling of the Hinting Session Status explained in Chapter 5. For example, this version was modified to take into account subdialogues, which were not part of the implicit student model, as they had not been formalised. The algorithm was in general augmented to accommodate possible student answers better, based on the data. This attempt was also complemented by our theoretical investigations that we explore in Chapter 2.

*Dialogue Management:* An extended analysis of general tutorial dialogue phenomena was based on the data from the experiment. The results of this analysis were used to define our dialogue-move taxonomy for tutorial dialogues (cf. Chapter 4). For example, the need to add an expanded task dimension was revealed, as we realised the level of complexity of handling both task and dialogue phenomena in order to adapt both for the final tutorial feedback.

All the above observations influenced the conceptualisation of the general architecture of the prototype tutorial dialogue system of the DIALOG project (Section 1.3.2) and of the architecture of `Menon` in particular. For example, defining a strategy in `Menon` where domain-independent tutorial feedback is handled and dividing substrategies into subdialogues and subtasks, were both instructed by our observations.

The particular ways in which we adapted and enhanced our formalisations were further informed by suggestions that our human tutor made and by our detailed pedagogical model that underpins our work psychologically. We explore the pedagogical model in Chapter 2. We also indicate the tutor's input where the relevant improvements are mentioned in the thesis.

## 1.5 Conclusion

The purpose of this thesis is to contribute to the research on automatic feedback for arbitrary problems and thus reduce the cost of pedagogical feedback and increase the number of tasks that can be included in an ITS for practice. By doing this, we hope to provide extra support for students, who might otherwise be at a disadvantage. The typical exposure of students to a few, at best, example problems in the classroom is a poor learning method [Sweller, 1989; Collins and Stevens, 1991]. This teaching method, vastly applied at schools worldwide, requires students to solve a larger number of problems at home, with feedback coming only much later than the attempt. The feedback mostly consists of a minimal marking of the homework by the tutor. Still, there are not enough human resources to tutor students in an interactive manner on solving problems, which is a superior method no matter what the particular teaching strategy used is (e.g. [Bloom, 1984; Cooper and Sweller, 1987; Corbett *et al.*, 1997]).

It is common practice in learning theories that learning is correlated with the amount of experience a student has in the domain [Eysenck and Keane, 2000]. VanLehn and colleagues [VanLehn *et al.*, 2005], for example, credited the success of their system largely to the fact that they implement many problems for the students to practice, as well as the style of tutoring they use, namely hints. They put the emphasis not on the quality of the hints so much, as to the fact that there are hints produced following certain educational principles for a great number of problems. On the other hand, they also point out that hard-coding hints, as they do, involves huge effort, time and human resources, as a constant collaboration with domain experts is necessary.

The approach to automating hints for arbitrary problems and solutions presented in this thesis is an attempt to tackle this problem. ITSs that teach well-defined domains, which is the most common case, are already dependent on formal representations of the domain knowledge and theorem provers for evaluating different solutions. Given that this infrastructure is required anyhow, the effort for structuring the domain in the way we suggest would be much less compared to authoring feedback for every task, every solution to that task, and every possible subpath or mistake in this solution. With regard to the Socratic teaching strategy that we implement, as this is largely domain independent, any adaptations to cater for the needs of other domains would be relevant to functions that deal with the domain-knowledge dimension of hints. The odds are that the quality of hints automatically produced by the system will not be as high as those produced by human tutors. However, producing hints for arbitrary problems gives students the possibility to practise on more problems and hence learn more. Moreover, the general advantage of a tutorial system over human tutors, namely that there is no upper time restriction, can only be taken advantage of if hinting is automated. Otherwise the feedback limitation restricts also the practising time. The automatic choice and the automatic content determination of hints undertaken in this thesis [Tsovaltzi *et al.*, 2004b], adds the possibility of providing feedback on arbitrary problems and solutions. This renders the hard-coding of feedback for every single proof unnecessary and reduces the cost of implementing feedback. Moreover, the adaptivity of hints in terms of NL generation makes up for at least some of the lost quality of hints in comparison to feedback that is pre-written and tailored to specific problems.

The definition of hints and the teaching strategy as a whole based on their constituents allows a later scientific approach to investigating learning. The different constituents can be manipulated separately and the study of many variables that influence learning can be operationalised for further empirical investigations. For instance, the pragmatic hints, or the meta-reasoning dimension, or a specific instructional point in the meta-reasoning dimension etc. can be switched off and the effectiveness of the resulting strategy can be compared to that of the Socratic teaching strategy as it stands. Such empirical studies can provide insight into the function of hints and into effective teaching strategies.

An evaluation that aimed at getting indicative feedback on the implementation of `Menon`'s Socratic teaching strategy gave very promising results. More specifically, we asked 5 evaluators to rate `Menon`'s Socratic teaching strategy as a whole, and individual instances of the automatic feedback produced by `Menon` in particular tutoring situations, and to compare it to equivalent feedback of the "winning" strategy in the Wizard-of-Oz study. The major finding of this evaluation was that `Menon`'s automatic feedback was considered better than the previous "winning" strategy's feedback that was produced by a human tutor. 4 out of 5 evaluators preferred Menon's teaching strategy as a whole. An evaluation of how Menon's overall strategy with specific reference to our global tutorial goals – defined in Chapter 2 – and Bloom's [Bloom, 1956] Affective and Cognitive levels gave good results, which means that the automatic feedback serves our tutorial goals to a large extent. Additionally, 55% of the evaluators' choices of individual instances of feedback favoured Menon's feedback to equivalent feedback of the previous "winning" strategy. The evaluation of this individual feedback instances with regard to our global tutorial goals was in total very good. These results can only be seen as a tendency, since the sample was small, but they show that the direction we have taken in the development of Menon is the right one.

# Chapter 2

# A Pedagogical Model for Tutoring

## 2.1 Introduction

This chapter presents and motivates our teaching model. It reviews the state of the art in the learning sciences, namely cognitive and experimental psychology, educational psychology and other scientific investigations in learning. The aim of the chapter is to make use of the prominent theoretical notion of schemata (cf. Sections 2.2.1 and 2.2.5) as well as the highly influential theories of cognitive load and motivation, and to derive a teaching model. The final outcome is the definition of the teaching model for the implementation in the tutorial manager `Menon`.

In this chapter, we also define our general tutorial goals. The global tutorial framework and the model, which we presuppose, set the context for our tutoring model. We then concentrate on the Socratic teaching model, as well as on the more specialised hinting process that we adopt in order to realise the Socratic model.

In summary, the teaching model defined here strikes a balance between (i) non-goal-specific tutoring, which allows students to build their own knowledge on existing structures and form helpful schemata, and (ii) making use of the tutor's expertise without super-imposing any cognitive model. Our general goals assume as a theoretical cognitive basis the following main themes: (a) motivation during tutoring, (b) assistance in the construction of cognitive structures (schemata) via heuristic instructions, (c) reduction of cognitive load and (d) promotion of implicit learning with moderate explicit learning.

## 2.2 Motivation of the Teaching Model

Teaching proofs has been part of school curricula worldwide for decades. Its value as a subject goes beyond the narrow goal of teaching students how to prove. Although the latter is a important aim in itself, the reason proofs have been taught for so long is that there is a recognised value in the fact that teaching proofs increases the students' ability for logical deduction and its application [Wu, 1996]. Wu [Wu, 2001] states that "it is well known that logical reasoning is synonymous with theorem proving. What is less well known is that logical reasoning is also the same as problem solving." [Wu, 2001] (p.3). The teaching model that we adopt is designed to enhance problem solving skills.

In the following sections we follow a thematic exposition of the prominent issues that need to be considered in choosing our teaching model. We look at the different issues always from the point of view of their impact on teaching and learning, even though they may have further cognitive and computational implications.

### 2.2.1 Schema Theory

The main cognitive theory that informs the pedagogical model discussed in this chapter is the schema theory. It was proposed as a complete theory by Rumelhart and Ortony [Rumelhart and Ortony, 1977; Rumelhart, 1980]. Although similar principles had been previously advocated by Minksy [Minsky, 1975] who used the term "frames" instead of schemata and Schank and Abelson [Schank and Abelson, 1977], who used the term "scripts", both frames and scripts aimed at explaining declarative representation of knowledge. *Schema theory*, on the contrary, supports that schemata are built actively by learners and any new information is assimilated in the structures of the existing schema, thus revising it. These revisions depend on the situations experienced and account for learning. Knowledge is represented in meaningful chunks and is constructed actively by the learner in networks of propositions. Wittgenstein has made a distinction between proposition and expression that clarifies this [Wittgenstein, 1975] (3:3111). According to him, *expression*, which is what schema theory really refers to, is the abstract form of all the propositions in which the expression can potentially occur and it is the carrier of meaning. Schemata have also been described as generalised, reapplicable descriptions of similar problems and their solution, e.g. [Cooper and Sweller, 1987]. An example of a schema application from [Sweller, 1989] (p.458), is the following. If the problem to solve is the equation (a + b)/c = d, an applicable schema would help to recognise this problem as one where the first step is to multiply both parts of the equation with the denominator c, in order to get rid of the fraction. Schemata capture this kind of reasoning in a step-wise manner.

Schema theory gave rise to artificial intelligence notions like *explanation-based learning* [Russell and Norvig, 2003]. They implemented the idea of generalising from examples and forming explicit rules for learning that are applied in new situations, which turned out to slow down the learning process. *Case-based* *reasoning* [Russell and Norvig, 2003], which is a successor of explanation-based learning, took these lessons into account. A case consists of the context, that is the world situation in which the problem occurred, the solution to that problem, and the state of the world after the case. Learning is based on experiencing different cases [Watson and Marir, 1994]. This approach offers itself to situated learning [Collins and Stevens, 1991], as it deals very well with the situation where the student is presented with the appropriate tasks (cases) to deal with at each point [Watson and Marir, 1994]. Since we are concerned in this thesis with promoting schema acquisition in human learners via the process of hinting, it is important to have a way to represent a schema in the form of its constituents, which are the elementary sub-schemata [Rumelhart, 1980]. In fact, the representation of cases is a matter of contention in the case-based reasoning community [Watson and Marir, 1994]. Furthermore, case-based reasoning does not provide a way of attacking this problem.

The definitions of schema used in experience-based learning and case-based reasoning proved to be too poor to capture the cognitive processes of understanding and learning that the abstract theoretical construct of the schema captures. A main reason for this inadequacy is that schema theory clearly stresses the need for a large exposure to varied learning situations before the schema acquisition can take place [Eysenck and Keane, 2000], whereas algorithmic applications are concerned with efficiency, and hence aim at generalisations from as few examples as possible using additional heuristics [Russell and Norvig, 2003]. Our aim is to lead students to acquiring such heuristics through the process of learning in a personalised manner. The acquisition of schemata is not observable as such, but the acquisition of the heuristics is, as one can observe the student student making steps that correspond to the ones included in the heuristics.

Later connectionist theories and models [Rumelhart *et al.*, 1986; Sun *et al.*, 1996; 2001], have gone beyond the somewhat static idea of schemata to an active organisation of knowledge that implies intractable forms of learning. The theory of schemata finds its application in tutoring in the convergence of instructional design and cognitive teaching models and in particular with regard to problem solving [Rumelhart and Ortony, 1977; Rumelhart, 1980; Price *et al.*, 1997; Widmayer, URL].

In our model, we propose to represent constituent meaningful parts of schemata and to use those as instructional points for the definition of the domain content of hints. Therefore, we do not claim to represent the complete schema. We only use the instructional points to help the student build a schema, even though we do not know which schema that is precisely. The reasoning here is analogous to that used in studies that seek to attribute the use of an expert-like method to solving problems to the acquisition of relevant schemata [Chi *et al.*, 1982; Lim and Moore, 2002]. They use the theoretical notion of a schema to explain how learning occurs, but are not interested in what this schema looks like.

### 2.2.2 Worked Examples vs. Problem Solving

In this section, we review studies on the two main approaches that have been proposed for teaching skill acquisition similar to proving: Worked examples and direct problem solving. We investigate their advantages and disadvantages for different educational purposes and choose the most appropriate for teaching proving and problem solving in the interactive phase that `Menon` implements.

Sweller and Cooper [Cooper and Sweller, 1987] as well as Owen and Sweller [Owen and Sweller, 1985] tested the use of worked examples with respect to learning gains. *Worked examples* are solved tasks, in our case examples of proofs, which the students study. The examples provide explanations of the problem solving steps. The comparison was a problem solving condition that asked students to solve new problems as a training method. In this condition no explanation was given and the only feedback was whether the solution was correct or wrong at the end. Sweller and Cooper found that high-level students with experience in mathematics when learning from worked examples were able to solve problems similar to the training problems better than students in the traditional problem-solving teaching condition. Sweller and Cooper were also able to support the hypothesis that the ability to transfer skills to different problems was higher in the worked examples condition and the time that students took to solve problems was less. The experimenters note, however, that the students in the traditional problem solving condition were of significantly lower-level and were taught for less time than the students in the worked-examples condition.

The most striking result of this research was that Sweller and Cooper established a connection between cognitive load and worked examples. *cognitive load theory* [Sweller, 1988] results from findings that *working memory* has limited capacity [Miller, 1956] and since information is first processed in working memory, learning is conditioned upon minimising the working load in short term memory. Making use of these concepts, Sweller and Cooper claimed that worked examples reduce the cognitive load that is predominant in problem solving, due to the fact that students do not have to be concerned with the application of domain rules that would impose extra load. They can, thus. concentrate better on the global solution strategy, learn better, and transfer what they learn to different problems.

Another reason for the reduction of the cognitive load and consequent better learning is that students do not have to concentrate on solving the problem as in the traditional goal-oriented teaching method. They can direct their attention to the different stages of problems and how they relate to the application of previously acquired knowledge. Over long learning periods, this way of learning enables useful acquisition of sophisticated schemata that include the automation of domain rules. Schemata are useful for evaluating how information can be used and they also facilitate the application of appropriate knowledge. This may be evidence for the superiority of instructional strategies targeting schema acquisition.

A later study by Sweller [Sweller, 1989] investigated in more detail what makes the teaching of science and mathematics effective. His line of reasoning was that problem solving is domain specific in nature, and students need to learn through domain-specific problem-solving schema acquisition, as they are already aware of general problem-solving techniques and how to apply them. His experimental evidence shifted the interest from the kind of teaching strategy to the presentation of the domain-specific material taught.

However, one could claim that the worked-example conditions in the various experiments were actually using problem-solving methods due to the way the strategies applied. These strategies involved students studying worked examples, asking questions, explaining the goals behind the choices made in the worked examples with the help of the tutor, and finally performing additional problem solving exercises, e.g. [Owen and Sweller, 1985; Cooper and Sweller, 1987]. This fact renders it difficult to draw conclusions on the worked-example teaching strategies. Therefore, it is better to argue with Sweller [Sweller, 1989] that what really matters is the exact way of presentation, rather than the distinction between worked-example vs. problem-solving strategy. Following Wilson's [Wilson and Cole, 1991] argument that all instruction is one way or the other embedded in some problem solving and that some domains by definition require problem solving for situated learning, problem solving appears to be a good environment for teaching how to prove.

Lim and colleagues [Lim *et al.*, 1996] performed a more detailed study on what enhances learning in particular individuals. They conducted a study in order to compare worked examples with problem solving. In the training phase of their experiment, they employed a non-traditional problem-solving method, that is a strategy that asks students to use the *forward technique* [Chi *et al.*, 1982] and just work from givens irrespective of a final goal[1]. They contrasted that to the traditional strategies that look at only one specific goal, which they used in the worked-examples condition. The participants in this condition saw worked examples of the solutions in the training phase. Both groups were then asked to solve goal-specific problems. They showed that participants in both conditions improved, but the *non-goal-specific* condition participants improved more, demonstrated more transfer skills, and maintenance, and were more efficient in problem-solving, although no help was provided other than correct-incorrect feedback. Echoing Sweller [Sweller, 1989], the experimenters attribute this difference to the fact that schema acquisition is hindered in the worked-examples condition, due to a split-attention effect that the worked-examples presentation induces. The *split-attention effect* in this case derived from separating the target problems from the example sources. In general this effect characterises the cognitive overload noticed when people have to process information from disparate sources [Sweller, 1988].

The presentation of worked examples and problem solving exercises one after the other, as used in the model by Lim and colleagues, was also investigated by Trafton and Reiser [Trafton and Reiser, 1993]. They criticised the worked examples studies for not testing the degree of transferability achieved through the particular teaching model. They, thus, tested the results that the dissociation

[1] This strategy will be further discussed in Sections 2.2.3 and 2.2.5.

of examples and problems brings about, by separating the examples from the equivalent target problems the students were presented during problem solving. They concluded that the effectiveness of studying worked examples reduces significantly when the examples are not available to the subjects directly before they are asked to solve each target problem. They put this effect down to the fact that students did not have the possibility to refer back to the examples, and conjectured that students could also not recall the examples from memory. They further showed that both worked examples and problem solving are important in tutoring. Examples should be presented first and followed by problem solving exercises.

McLaren and colleagues [McLaren *et al.*, 2008] summarised the results of their own studies as well of other similar studies that investigated the interleaving of worked examples and problem solving as a means to combine their strenghts and effectiveness. They viewed worked examples and problem solving as methods that provide different degrees of assistance to the student. They can be split themselves into low and high-assistance methods according to whether explanations of tutoring is offerred along with worked examples of problem solving, respectively. This gives rise to the following continuum from less to more assistance: plain problem solving – tutored problem solving – worked examples with explanations – plain worked examples. Their research question was which combination of these methods yields the better learning results. Their own studies on Stoichiometry concentrated on comparing two conditions: one using worked-examples followed by multiple-choice questions that elicit self-explanation, and one using a tutoring system that provides hints on request and error messages. Both conditions learned but there was no difference in the learning effects. However, the worked-examples condition was more efficient. In general, McLaren and colleagues concluded that although no one method or combination has been consistently shown to be superior accorss studies, there is a tendency that the combinations in the middle of the assistance continuum produce better results.

Delclos and Harrington [Delclos and Harrington, 1991] came to similar conclusion in terms of the use of worked examples in combination with problem solving. They found evidence that students must be given the opportunity to make use of their taught knowledge in a real problem-solving situation, which constitutes the experience on which schema acquisition is based. The procedural nature of this type of experience is what promotes schema acquisition. This opportunity should be accompanied by instruction both before and during problem-solving exercises and should partly concentrate on enhancing reflection on the student's own strategies. They, thus, made a shift towards the kind of instruction used.

In the same direction, Ian Robertson [Robertson, 2000] studied the effect of particular styles of presentations of worked examples. He analysed algebra word problems in order to investigate transfer and predict problematic areas depending on the previous presentation style, even though the worked examples were available during the problem solving exercises. Like [Sweller, 1989], he does not argue for or against worked examples vs. problem solving, but he suggests specific ways of presentation. Namely, he found that transfer increases with general abstract instructions on the procedure, instead of on specific inferences relevant only to the examples presented. He claims that this allows students to generalise from the studied examples and transfer the acquired knowledge to near or distant variants.

Mathematicians have doubted the benefit of studying worked examples in the domain of theorem proving. Wu [Wu, 2001], argued that there are elements in the proving process that are not captured in the static representation of the same proof. For example, he claims that the order in which the proof is presented for better understanding does not necessarily match the order in which the step is arrived at. That may cause students to learn a counter-productive way of proving, which hampers rather than assists the proving process. Worked examples constitute static representations by definition, and are hence vulnerable to this criticism. Wu does, however, agree with Trafton and Reiser [Trafton and Reiser, 1993], to the extent that he advocates the use of many examples in a constructive way and as necessary, before students are asked to solve problems themselves.

The above exposition indicated that the two approaches to tutoring problem solving, worked examples and direct problem solving, are not so much competitors as appropriate for different phases of tutoring and different tutorial goals. We choose problem solving as the context of the interactive tutoring sessions for proving that we implement for two main reasons: first, because it seems to be more appropriate for teaching the procedural nature of proving, second, because it is a better platform for exposing students to the kind of experience that promotes schema acquisition. With regard to worked examples, we assume the students' prior exposure to them, although we are not concerned with modelling the tutoring phases in which they would be used. Moreover, we also take into account the observations on the specific ways of instruction that make problem solving as a tutoring method more effective, which we discuss more extensively in the next section.

### 2.2.3 Kinds of Problem Solving

In this section we consider the use of non-goal-specific problem solving in order to support schema acquisition and resolve the cognitive load drawback of traditional goal-specific problem solving.

Sweller [Sweller, 1989] examined the source of the success of the worked example strategy that he had earlier proposed. He showed that problem solving becomes powerful when a shift from goal orientation to learning orientation is made. The theoretical justification for these findings is that the aim of solving the problem in goal-oriented approaches inflicts extra, unnecessary cognitive load that is counter-productive to learning. This load is lifted by the shift to non-goal-specific problem solving.

Lim and Moore [Lim *et al.*, 1996; Lim and Moore, 2002], came to similar conclusions with regard to the factors that facilitate learning in general, and schema acquisition in particular. As we saw, they compared the learning effects between a worked-examples instruction strategy and a non-goal-specific problem solving instruction strategy in geometry. More specifically, the non-goal-specific strategy, this asked students to calculate as many angles as possible, as opposed to calculating a particular angle that, which was demonstrated by use of a worked example. They found that non-goal-specific problem solving is superior to worked examples that involve a specific goal in the interactive phase. The group of students tutored by use of the non-goal-specific strategy solved more problems, faster, with less errors, using a technique more similar to the experts' technique [Chi *et al.*, 1982]. These subjects were also better able to transfer their skills to other problems, as well as maintain their skills over time. This experiment indicates certain aspects of the non-goal-specific problem solving that are relevant to the interactive tutoring phase that we want to model.

At first glance, it seems somewhat problematic to define a non-goal-specific teaching model for proving, that is for teaching natural deduction. In particular, it might seem like the learning goal coincides with the proof goal. In other words, part of the tutorial goal is that the student becomes able at some point to arrive at a given goal based on some specific givens and a knowledge base. This is the aim of our module. However, there is a clear distinction between learning how to prove in general and learning how to prove a particular problem. A student might never be able to solve a particular problem, but still acquire valuable knowledge on proving.

We adapt a non-goal-specific teaching model that allows the student to find any route towards the proof goal. Students can thereby learn to work with the constituents of proving, rules, methods, techniques etc., irrespective of how, or if at all the goal is reached. The type of the instruction we use is, therefore, non-goal-specific and students can draw as many inferences from the givens as possible and they can rely on the computer to tell them which ones are applicable or to guide them to applicable ones. Using non-goal-specific problem solving for the interactive phase, which `Menon` models, appears to be the state-of-the-art answer to what has been often perceived as an incompatibility between attention and cognitive load theory, on the one hand, and constructivism and schema theory, on the other hand [Goldman, 1991; Chandler and Sweller, 1991]. Such views have criticised cognitive load theory for not taking into account the approach of constructivism and schema theory that is in favour of some cognitive load to allow the construction of schemata. However, cognitive load theory argues for eliminating unnecessary or extraneous cognitive load in order to promote schema acquisition and extraneous cognitive load is defined as everything that is does not add to schema acquisition [Sweller and Chandler, 1991]. The apparent incompatibility of the two theories dissolves when one combines problem solving, which creates the situation for schema acquisition, and non-goal-specific instruction, which is a means of eliminating the extraneous cognitive load.

### 2.2.4 The Motivation Theory Standpoint

In this section we look into arguments from motivation theory [Keller, 1987; Weiner, 1992; de Vicente and Pain, 1998] to compliment our teaching model.

We devote a section to it, as motivation theory is a field in and of itself, which requires special attention. Note, nonetheless, that this work concerns itself with motivation to the extent that it can be served via our primery goal; automating hinting. This is consistent with the view that sees motivation as a meta-model that guides specific instructional decisions.

Motivation theory does not deal with what or how much students learn, which is what we have been looking at so far in this chapter. Rather, it deals with what students are willing to do and how much effort they are willing to put into learning. The aim is to increase motivation and thereby increase students' interest and their ambition to learn. This can be done in different ways, for instance, by presenting the learning task as a game to make it appear more interesting, or by informing the teaching strategy accordingly. Here we are interested in how the teaching strategy functions as a medium for motivation purposes. Motivation influences the student's ability indirectly to the extent that the learning effects are dependent on the amount of time the student spends learning and the degree of attention applied. These two variables are just indicative, and they do not cover the range of psychological ramifications that are captured by the term motivation. They are mentioned here as the ones that can more easily be measured and also correlated to learning gains. There is, however, research in the direction of motivation diagnosis [Rebolledo-Méndez, 2003; Mavrikis *et al.*, 2003].

Let us now look at the principles of motivation theory. The father of motivation theory is Abraham Maslow [Maslow, 1943], who was the first to state in this context the interconnection between bodily and intellectual needs. He sketched the original picture that depicts the role motivation plays in humans. His theory, pioneering as it was, was too abstract with many vaguely defined concepts and processes. Consequently, it cannot be operationalised for the purposes of this thesis, that is for the definition and implementation of a tutoring model. Therefore, we concentrate on later research, and more specifically on the theory of J. Keller [Keller, 1987]. Keller used previous motivation theories and combined them into a unified one with the aim of applying them to instructional strategies. Hence, his research is most appropriate for our purposes. According to Keller, there are four main levels of motivation: (i) attention, (ii) relevance, (iii) confidence, and (iv) satisfaction. We briefly look into them and what they involve, and we propose a way of incorporating them in our instructional strategy.

*Attention* describes the degree and the persistence of engagement of the student at any time. The feature of instructional strategies that can help maintain the student's attention is providing at each point the right amount of information that would help the student with the task without giving away too much. Giving hints is a way to realise this, instead of the traditional didactic explanations that tend to give too much information at once and be long. Such explanations are, therefore, counter-productive for maintaining or increasing attention. Hiroyoshi Watanabe [Watanabe *et al.*, 2003] in his work with intelligent tutoring systems has found that students learn better via short hints than detailed explanations, which tend to be too long and boring for them and fail to maintain their attention. Ashley and colleagues [Ashley *et al.*, 2002] also reached conclusions supportive of this view. They called their strategy, that is analogous to what we call `Socratic` strategy "dialectic". They observed an increased level of involvement in the condition using the dialectic strategy, in contrast to the compare group where didactic explanations were used. Although involvement is not the same as attention, it is an indicator of increased attention.

Moreover, according to motivation theory, people mostly pay attention to things they already know something about [Keller, 1987]. The use of hints in itself as well as trying to give hints for the solution that the student has in mind, raise the student's attention. Problem solving itself has an advantage in this regard. By its nature it tends to involve the learner in the task as a contributor. The use of questions in the form of hints increases the engagement of the learner [Keller, 1987].

*Relevance* refers to the students feeling that the specific instructions that they get cater for their personal needs and goals. It mostly relates to the feeling of achievement in terms of meeting goals and succeeding in things. *Achievement* describes the desire to overcome obstacles, accomplish goals and succeed in tasks. It also relates to the feeling of power; that is, the feeling that the student influences and has control over the task. This latter point can be a side-effect of computer tutoring systems, since the user already feels superior in many senses and in control by default. The feeling of achievement can be realised by the system trying to make use of everything in the student's attempts that is useful for any proof. This presupposes looking for necessary key points in the student's input that are useful for the proof, and using them to make the students feel that they are contributing to the task.

Exactly those features also contribute to creating *confidence*, that is, giving the students the feeling that they are managing well and that they are going to succeed in the task. This level of motivation speaks against goal-specific strategies, which often deprive students of the feeling of success, as students are more likely to fail the more restricted the goal is. Both disallowing the student to find just any proof they could, and asking them to find a better proof afterwards can have the effect of learned helplessness. *Learned helplessness* describes the result of dissatisfaction deriving from such constant attempts to succeed and finding success almost impossible. Learned helplessness reduces effort. Confidence is also related to how much the tutor believes that the student will accomplish the task. Although this can be read by the student in the tutor's instruction choices, additional emphasis can be put on the positive aspects of the student's input and extra encouragement in natural language can be provided as part of the Socratic instructional strategy, in order to increase the student's perception of confidence building features in the behaviour of the tutor. Ashley and colleagues [Ashley *et al.*, 2002], for example, found that students in the dialectical (`Socratic`) condition feel less judged for their mistakes, which can explain why they have high levels of confidence.

Explicit encouragement from the tutor caters also for *satisfaction*, which is the fourth and last of the basic levels of motivation. Satisfaction can be effected by extrinsic motivation, which is possible to manipulate in an instructional

`Socratic` tutoring dialogue system. The most important form of satisfaction, though, comes from intrinsic motivation and is at the same time difficult to observe and to manipulate. Intrinsic motivation is promoted when the other three levels of motivation are met.

Contrary to human tutors, though, computers lack any ability to discipline the student in the sense of having them pay attention and attending to the task. Students are aware that they are in control. Superficial as this might be for the purposes of motivation, it can turn out to be detrimental for any intelligent tutoring system that does not cater for motivation. Namely, it can result in the student simply switching the computer off. For this reason, along with aiming at the cognitive state of students, we also try to take motivational aspects into account in the particular model of the `Socratic` style that we adopt, although it is not in the focus of this research and we keep in mind the special aspects that feature in human-computer interaction, as opposed to human-human interaction [Shechtman and Horowitz, 2003].

Various researchers are also investigating politeness as the counterpart of motivation in terms of language use. [McLaren *et al.*, 2007] have integrated politeness in an existing e-Learning system for Stoichiometry. They conducted in-vivo empirical studies comparing polite feedback vs. the standard feedback of the system. They found a trend for better learning effects when polite feedback was provided, although the particular integration of politeness in their system did not show significant results. [Porayska-Pomsta and Mellish, 2004; Porayska-Pomsta and Pain, 2004; Mayer *et al.*, 2006] investigate models of NL generation to accommodate politeness in the feedback provided and in the way it is presented to the student. *Autonomy*, letting the student do as much of the work as possible, and *approval*, providing as much positive feedback as possible, are employed by Porayska-Pomsta and colleagues. Although this work concentrates on NL generation, it is relevant to our work first, in that we recognise autonomy and approval as the possible instigators of attention, relevance, confidence and satisfaction, and second, in that we believe that the use of automatic NL tutorial dialogue can incorporate and promote such aspects of affect.

### 2.2.5 Non-goal-specific `Socratic` Teaching

The term *Socratic* signifies any teaching strategy where a dialectic tutoring style is used that aims at eliciting information from the students and promotes active learning, e.g. [Stevens and Collins, 1977; Rosé *et al.*, August 2001]. `Socratic` strategies emphasise the need for instruction and for monitoring the learning process. For instance, the snapshot from the example that we saw in Section 1.3.4 shows how the tutor directs the student to finding the next proof step by trying to elicit the relevant aspects of it from the student.

...

**T2:** (`encourage`) Great! (`signal-pa`) You're on a good track. (`initiate-subtask-proof-step-meta-reas`) We're taking it from the start. So, go ahead and (`elicit-prem-conc`) find what is assumed and what you have to prove.

**S2:** (`correct`) I have to prove that if $A \subseteq K(B)$, then $B \subseteq K(A)$, and $A \subseteq K(B)$ is given.

**T3:** (`signal-accept`) Correct! (`elicit-specific-method`) Now, how can you manipulate the expression to prove what you want?

**S3:** (`correct`) I have to simplify what we are trying to prove.

**T4:** (`signal-accept`) Correct! (`close-subtask-proof-step-meta-reas`) OK. (`initiate-subtask-rel-con-meta-reas`) Let's see, then. (`elic-meta-reas-rel-con`) Try to find something in the expression that would help you simplify the problem.

**S4:** (`correct`) Do you mean the if-then?

The motivation for using the *`Socratic`* teaching model for our purposes is discussed in this section. More specifically, we present studies that examined what can be improved in non-goal-specific problem solving in order to further reduce cognitive load. In this context, we also look into promoting schema acquisition in more detail.

As we mentioned in Section 2.2.3, Lim and Moore's [Lim and Moore, 2002] experiments aimed at measuring the acquisition of schemata. We now turn to the aspects of their study that specifically refer to schema acquisition. In particular, Lim and Moore argued, against Cooper and Sweller, that the advantage of worked examples is that they provide enough information to the students in order to avoid cognitive load and allow the formation of schemata. They hypothesised that using appropriate instructions for students during non-goal-specific problem solving would result in even higher cognitive gains. They analysed, among other factors, the students' responses during learning, comparing two groups: one group with additional instructions and one without. Their experiments confirm their hypothesis, showing that better schema acquisition was possible for the group that was instructed through leading questions during problem solving.

Wilson and Cole [Wilson and Cole, 1991] pointed out the connection between cognitive teaching models and the more established at the time instructional theory. They provided an argument for combining cognitive teaching models with instructional methods. *Instructional theory* supports providing heuristically based chunks of instruction during the learning process. Instruction is given in an incremental manner and in a simple-to-complex progression. The basis is behaviouristic, connecting learning goals with teaching strategies, but leaving out how the goals are realised cognitively. In the cognitive models tradition, on the other hand, instructions are meant to accompany exposure to experiences, or better be part of the experience every time. The experience itself is what gives rise to schema acquisition and learning [Delclos and Harrington, 1991; Driscoll, 1994].

Delclos and Harrington [Delclos and Harrington, 1991] also tested the effects of *monitored learning*, which promotes student control. They asked subjects to read preliminary domain instructions and evaluated the learning gains among three groups: one that received no further training, one that was left to self-monitored problem-solving training and one that received monitored problem solving. The monitored group received additional instructions during training, which encouraged them to monitor their problem solving. The results made evident the superiority of the combined training and monitored group. They therefore argued for the importance of instructions that promote strategy monitoring in problem solving.

Furthermore, a comparative study [Rosé *et al.*, August 2001] for the effects of traditional didactic teaching strategy vs. the more dialectically oriented Socratic one, which provides scaffolding instructions, has indicated that students under the Socratic condition learned more. Notably, the study investigated dialogue interactions between the tutor and the student in natural language. Although the two teaching strategies were not formalised for the study, there is an obvious difference in the quantity and the quality of feedback in the two conditions observable in the corpus collected in the study [Moore, 2000][2].

There are a number of further advantages to a ***Socratic*** instructional tutoring model. First, it can simulate the reduction of cognitive load, as recommended by the work we already looked at in this section due to its controlled instructive nature. Second, it takes advantage of the beneficial use of natural language in tutoring, as it has been argued in the literature, e.g. [Moore, 1993], in the form of hints. A good example of this is the automatic use of appropriate discourse markers, such as *"as we have seen before"*, *"as you know"*, *"moreover"* etc., which make information processing easier, hence minimising cognitive load. The flexibility that natural language, and with it dialogue, allows for simulating some of the motivation aspects in tutoring, which we discuss in the following section, which are difficult to manipulate otherwise. Third, the dialectic nature of instructional tutoring opens up an area of manipulation of other cognitive facets that tutoring should take into account, namely those that can be handled by natural argument [Reed and Grasso, 2004]. For instance, we saw how the use of natural language allows manipulating discourse coherence and with it task coherence, and how the different factors of tutoring feedback are combined in the instructions provided by `Menon` in the example in Section 1.3.4.

### 2.2.6 Forward reasoning instructions

So far we have looked into general teaching strategy issues. We now go into more detail and support the use of instructions that enhance schema acquisition by providing step-wise hints. More specifically, we look into what the order and the content of hints should be, in terms of the abstract problem-solving technique that the students are guided to. For that we examine two more major issues in teaching how to prove; teaching forward vs. backward proving, and teaching declarative vs. procedural knowledge. This section tackles instructions on forward vs. backward proving.

An early study by Chi and colleagues [Chi *et al.*, 1982] compared the problem solving techniques used by novices to the ones used by experts. They found

[2] In the first phase of our formalisation of the tutoring model, we used this corpus as an empirical basis for our work (cf. Chapter 1).

that experts use a *forward-reasoning technique* for easy problems, moving progressively from one goal to the next, until they have reached the end goal with minimum manoeuvring. On the other hand, novices working on the same problems seem to oscillate between the current stage and the goal, and use many unordered steps, which they then re-order and bring together, as they approach the final goal. This is called the *means-end technique*. Their research suggested that the reason behind this difference is the possession of schemata, in the case of experts, and the corresponding lack of them, in the case of novices. The reasoning is that when a schema exists, the next step to be taken is encoded as part of the schema, which enables the possessor of the schema to move forward taking one step after the other in an ordered manner. By the same token, when a schema is not available, a person solving a problem must reason backwards from the goal, in order to decide which step to take next on the basis of what would help with reaching this goal.

Matsuda and VanLehn [Matsuda and VanLehn, 2003] looked into the hinting strategies used by humans, and the behaviour of students when solving geometry proof problems. They also observed that students did not use a consistent forward or *backward*, working from the goal, proving style, that is, they used the mixed directionality of the kind that Chi and colleagues have found to be characteristic of novices. That lead them to speculate that it would be beneficial if the tutor adapted the hints to this opportunistic technique and, in effect, used no expert model to guide them. In a later study [Matsuda and VanLehn, 2005] they compared two strategies, each concentrating on one of the aspects of the early study. Namely, one was teaching forward reasoning and the other backward reasoning. Both of these strategies are implemented in the Advanced Geometry Tutor intelligent tutoring system, e.g., [Matsuda and VanLehn, 2005]. They found that the forward chaining teaching strategy effected more learning gains than the backward chaining teaching strategy. Data analysis specified the disadvantage of the backward chaining strategy to the difficulty faced by the students to set subgoals necessary for backward reasoning. They interpreted those results as consistent with previous studies that have found that novices find backward reasoning difficult.

We will now look at research that addressed the question of directionality as part of a teaching strategy aiming at learning, on the one hand, and as part of the reasoning employed normally by problem solvers irrespective of instruction, on the other.

In the mathematics educators forum Wu [Wu, 1996; 2001] has argued that when teaching how to prove, a model must be provided. This model should assist the learning process with specific reference to the directionality of a proof. More specifically, he states that students should be taught appropriate heuristics for finding a proof, but should be made aware of the difference between heuristics and proof.

Cognitive studies have also concerned themselves a lot with the issue of directionality. Owen and Sweller [Owen and Sweller, 1985] claim that the means-end technique identified first by [Greeno, 1978; Chi *et al.*, 1982], although it achieves the goal of solving the problem, hampers schema acquisition and, hence, learning. The speculated reason is the useless extra cognitive load posed by the means-end technique. They were able to show, by a series of experiments in trigonometry [Owen and Sweller, 1985], that leading subjects to a representation of key points in the problem solving helped them reduce errors both during treatment and subsequently. Subjects also demonstrated better transfer of learned skills, arguably due to the creation of helpful, expert-like schemata, which allowed them to apply the acquired knowledge to new situations.

Lim and Moore [Lim and Moore, 2002] also used a forward-looking non-goal-specific problem-solving condition in their experiments. They tested specifically directionality, among other things. Their findings showed that subjects under this condition improved in the application of forward directionality, that is, they used very early on the expert, forward technique. On the contrary, subjects on the non-instructional condition mostly did not manage to achieve the required directionality at all. Moreover, there was a correlation between the number of errors and the directionality used. Subjects using forward directionality made fewer errors and were in general better at the performance of the task. Additionally, they maintained their problem solving abilities over time to a higher degree than the non-instructed subjects.

Sweller [Sweller, 1989] observed that the means-end technique used by novices, which also uses backward reasoning, is indispensable for problem solving. Still, it is all the same inappropriate as a tutoring model for instruction. He argues that the extra cognitive load imposed by such a technique hinders learning rather than assisting it. This cognitive load derives from having to split one's attention between the goal, the givens, and the relation between them, while this is the same for any subgoals. In tutoring proving, where the goal is necessarily part of the problem, non-goal-specific tutoring is only feasible in the sense of allowing any possible solution. To alleviate extra cognitive load imposed by split attention, extra help can be provided that indicates when a backward step (starting from the goal) should be applied. Then the student does not have to constantly bear in mind the goal.

Moreover, [Lim *et al.*, 1996] observed a similar phenomenon, while experimenting primarily with non-goal-specific vs. worked-example instruction strategies. They could not show a clear correlation between the proving technique used and learning. Some students who used the forward technique did not learn or maintain any learning, while other students who used mixed directionality both learned and maintained learning gains. Lim and colleagues could not account for this phenomenon based on existing learning models. However, the phenomenon is possible to explain in the light of other research results. First, no instructions towards a forward technique were used that would help the students who did not have a schema yet to acquire one [Sweller, 1989]. Second, the students were deprived of the default means-end technique, due to the application of the non-goal-specific strategy that did not provide the student with an end [Greeno, 1978]. The combination of these two drawbacks seems to be what prevented learning for certain subjects. [Greeno, 1978] showed the means-end strategy can be applied more successfully when hints for one of the intermediate proof steps are provided.

In our own Wizard-of-Oz experiment, all students could easily learn to use forward steps, even if they were not sure which rule to use. On the contrary, they had great difficulty with backward steps. This was observed in 17 out of 19 students across conditions. It was very puzzling for the students and the main time-consuming aspect of the Socratic teaching strategy used in the experiments where the forward step was requested. One could hypothesise that the extra cognitive load imposed by insisting on the students learning the backward reasoning in the Socratic condition hindered the learning process [Sweller, 1989]. In the context of the Geometry Tutor [Anderson *et al.*, 1993] as well, explicit tutoring of backward steps had to be taken out of the system, because it was problematic. Although the system provided a useful diagrammatic display, the students were confused. They tried to set subgoals for proving something that the goal implied, instead of ones that implied the goal. Another alternative to standard backward steps is making use of what is present in the given situation, thus saving on memory storage, choosing applicable operators with reference to the situation only, and combining operations based on schemata [Greeno, 1978].

In summary, the research cited here shows that the two techniques are complementary for learning. However, if the aim of instruction is to turn novices into experts, then the forward technique should be promoted. Therefore, in our tutoring model we favour forward reasoning as much as possible, but instruct on the use of backward steps when these are necessary in order to support students who use the means-ends technique. We, thus, hope to permit the creation of schemata that will eventually arise from the exposure of the learner to problem solving situations.

An additional issue about the style of tutoring, which becomes relevant at this point, concerns the choice between delayed and immediate feedback. Both cognitive load and the forward directionality argue for immediate feedback. Delayed feedback goes against these principles, as the student is left without help for some time, while moving without orientation. This creates unnecessary cognitive load and does not take advantage of instructing the student to use the more advantageous forward directionality. Empirical evidence to the support of immediate feedback comes, for example, from Person and colleagues [Person and Graesser, 2003], who observed a preference for it in human tutoring. Additionally, Freedman and colleagues [Freedman *et al.*, 1998] found that local evaluation is more important for choice of feedback. This finding also points to producing immediate feedback, as any local evaluation is rendered useless otherwise.

Moreover, findings from the open model community[3] support that reflection is at most promoted with the interaction of student and tutor. A study by Zapata-Rivera and Greer [Zapata-Rivera and Greer, 2003] examined the effect on reflection, by use of a taxonomy of different reflective acts. They found that deeper reflection is evidenced when immediate feedback is given.

Bearing in mind the issues discussed and given the nature of proving that

[3]The main philosophy of this community is that reflection is promoted when the student model maintained by the system is made available to the student herself, e.g. [Dimitrova *et al.*, 1999].

necessarily involves backward steps, we suggest the use of immediate feedback to help the student in the relevant situations. We also set the goal of concentrating on elements of knowledge that are present and can be referred to, and develop hinting around them.

### 2.2.7 Declarative vs. procedural learning: Effects of self-explanation and meta-reasoning instructions

In this section, we discuss declarative and procedural knowledge and their connection to instruction. We look into the connection between these two kinds of knowledge processing with the use of self-explanation and meta-reasoning in instruction. We identify advantages in both and propose ways of integrating them in our tutoring model so that procedural knowledge is the goal and declarative knowledge is used as a means to this goal.

In artificial intelligence, declarative knowledge has been used to mean knowledge encoded explicitly as facts. This is opposed to procedural knowledge, which is a direct representation of behaviour [Russell and Norvig, 2003]. Represented facts are declarative knowledge, whereas inference rules that allow further knowledge to be induced from those facts are procedural knowledge. In psychology, declarative knowledge is also defined as factual knowledge that is represented explicitly. It is manifested largely in the ability to recollect the facts causing the knowledge. Procedural knowledge, in contrast, is knowledge represented implicitly without conscious awareness. It is manifested in skilled performance. Declarative knowledge creates reference points that serve as points of attentional expectancies. Attention, in turn, advances procedural knowledge. However, declarative and procedural knowledge are dissociated, as one is not necessary for the acquisition of the other [Willingham *et al.*, 1989]. Although there is the general potential for transforming knowledge of one form into the other, not all implicit knowledge can be brought to consciousness and made explicit.

In the following research review, we preserve the preferred terminology of the researchers each time, but recognise a connection between declarative and explicit, on the one hand, and procedural and implicit, on the other.

We will now look at studies that investigated different models of learning, with particular emphasis on the distinction between targeting declarative or procedural, non-conscious knowledge acquisition. We will use this as a basis for justifying our only moderate use of self-explanation, a notion that has been widely adapted in the field of intelligent tutoring systems (e.g., [Person *et al.*, 2000; Conati and VanLehn, 1999; Aleven and Koedinger, 2000a]). The goal of self-explanation can be described as prompting students to articulate their thinking in order to transform implicit knowledge into explicit [Collins, 1991].

The highly influential study by Chi and colleagues [Chi *et al.*, 1989] supported the notion that self-explanation enhances learning. This was based on discovering that better human problem solvers can better explain themselves, that is explain what they are doing and why [Chi *et al.*, 1982]. However, there can be a number of reasons behind this observation. The fact that some people are better problem solvers might mean that they did at some point possess the declarative knowledge (or similar declarative knowledge) that self-explanation presupposes. This knowledge they can now retrieve from memory. Most importantly their high level in problem solving means that they can afford to self-explain, as they do not bear a heavy cognitive load. This kind of cognitive load might be the reason that disallows self-explanation abilities in less capable problem solvers.

Researchers have also looked into the distinction between acquiring procedural vs. declarative knowledge. The early hybrid model CLARION built by Sun and colleagues [Sun *et al.*, 1996] was designed to learn skills from procedural knowledge, without presupposing any declarative knowledge. CLARION unifies connectionist, symbolic, and reinforcement learning. It succeeded in demonstrating that this bottom-up approach facilitates transferability and even helps the learning process on-line. Another more elaborate study by the same group [Sun *et al.*, 2001], which was based on an advanced version of CLARION, modelled again a bottom-up approach to learning. Their architecture promotes bottom-up skill learning in reactive sequential decision tasks, whose nature is similar to problem solving. The agent learns in a sequence of interactions with the world, without the use of preconceived declarative concepts and knowledge on how to perform a task. Procedural skills and high-level knowledge are acquired via this kind of learning process, with declarative knowledge rising from the procedural knowledge acquired.

This learning model takes a counter view to Anderson's more conservative learning model [Anderson, 1993], as well as to the models reviewed by VanLehn and his colleagues [VanLehn *et al.*, 1992; VanLehn, 1996], which report on domains and specific knowledge that are suited for the use of top-down learning, and do not adapt as well to implicit learning as proving does. Wu captures this procedural nature of proving in the concise utterance: "... no intuition, no proof" [Wu, 2001] (p.31), intuition being nothing else but what is not declaratively available. The claim of the supporters of learning based on declarative knowledge is that procedural knowledge can arise from declarative knowledge [Anderson, 1993]. VanLehn and colleagues explicitly claim that self-explanation facilitates learning, as it evokes the formation of declarative knowledge. Sun and colleagues' [Sun *et al.*, 2001] more radical learning model further shows that bottom-up learning in synergy with moderate top-down learning is preferable, as it promotes transferability of the learned skill. It also has been shown to occur naturally in tasks that are rule based, like mathematics [Mathews *et al.*, 1989]. However, learning is potentially slower, because procedural knowledge cannot be learned in one incidence. The repetition of similar tasks is an absolute prerequisite for its construction.

Other researchers have also posited views in favour of de-emphasising forcing students to acquire declarative knowledge. Berry and Broadbent [Berry and Broadbent, 1984], for instance, argued for the dissociation between learning and awareness. They found that subjects who could answer questions demonstrating explicit knowledge did worse at their tasks. They suggested that in order to have a learning effect, subjects have to be guided on the content of verbalisation and not be asked to verbalise their thoughts. Only then can learning occur with subsequent requests for verbalisation. They recommend verbal instructions for achieving this. In later experiments [Berry and Broadbent, 1988] demonstrated that forcing learners to explicitly extract verbalisable rules while learning can impair performance on tasks as well as transfer. This is conditioned upon, first, the degree of saliency of the presentation of the rules to be learned, second, specifying relevant points that make the rules more salient. Since implicit and explicit learning can be operationalised this way, they claim that they are not mutually exclusive but work in parallel. Based on the degree of saliency of rules in a task, which may vary a lot in complex tasks, one kind of learning will prevail over the other. This point on explicit and implicit knowledge working together also fits in well with the connection found between attentional expectancies build by declarative (explicit) knowledge that assist procedural (implicit) knowledge [Willingham *et al.*, 1989].

More directly, Lewicki and his colleagues [Lewicki *et al.*, 1992] have vigorously supported the promotion of implicit, non-conscious learning. They argued that too much declarative knowledge can actually hinder learning of rules, and that non-implicit learning is much more efficient and structurally more sophisticated than consciously controlled learning. It is nonetheless intractable and cannot be formalised, as the human consciousness is unable to handle the complexity in which this kind of learning and the derived knowledge organises itself. That, in effect, has three consequences. First, during problem solving, a lot of knowledge is formed unconsciously in an implicit manner. Second, it is impossible for students to turn all implicit, procedural knowledge that they potentially acquire, into declarative knowledge [Mathews *et al.*, 1989]. Third, forcing students to report on the knowledge that they have acquired in the self-explanation fashion puts additional cognitive load on them, which hinders non-conscious processes. Moreover, it directs the students' attention to the consciously tractable knowledge, which Lewicki and colleagues [Lewicki *et al.*, 1992] claim to be inferior. Thus, an excessive demand for self-explanation can be counter-productive as far as learning skills is concerned.

Conati and VanLehn [Conati and VanLehn, 1999] have empirically observed that students are not inclined to generate goal-oriented explanations, which generally builds highly transferable skills. Moreover, studies in the Intelligent Tutoring Systems community, e.g. [Heffernan and Koedinger, 1998] have actually shown that it is more difficult for students to explain verbally what they want to do, rather than do it. This was shown for algebra problems, where verbalising a solution is more difficult than performing the solution in algebra.

Such observations concur with the earlier conjecture of Stanley and colleagues [Stanley *et al.*, 1989], who interpreted their experiment results as evidence for the indispensability of implicit learning. Their experiments gave evidence that learning on complicated tasks takes place implicitly until a very high level of expertise is reached. Only after reaching this level are learners capable of verbalising the acquired knowledge, possibly due to already complex forms of mental representations of the knowledge. They concluded that implicit processing is essential to acquiring the knowledge, and that short verbalisations or verbal instructions assist the learning process when based on heuristics relating to the task.

For our purposes, the conclusions on implicit and explicit learning mean that forcing students to verbalise their actions is not appropriate for a task such as proving, where procedural learning is the aim, because it fortifies top-down processes at the expense of the bottom-up processes required for proving. However, moderate request for verbalisation is beneficial. More specifically, Sun and colleagues [Sun *et al.*, 2001] arue that the heuristic process used for skill performance must be both implicit and explicit. It must be implicit in "making decisions based on current information in accordance with the 'policy' that implicitly takes into account future steps"(p.35). This speaks for schema promoting hints. "Explicit" refers to a more declarative approach. This can be done by explaining the meta-reasoning behind the heuristics applied during the promotion of skill learning in a rule fashion. This is also instructed for our domain by Wu [Wu, 2001; 1996] who is in favour of making students aware of the logical reasoning that informs proving. Moreover, Schoenfeld and Herrmann [Schoenfeld and Herrmann, 1982] also support the need to provide instructions relevant to the meta-reasoning. Their research concentrated on the differences of the structures of knowledge in novices and experts. They showed that experts rely more on deeper understanding of the principles of the domain, that is the meta-reasoning, rather than surface characteristics like words in the problem or syntax of the mathematical expression, in the case of mathematics, which is typical of novices. On the other hand, meta-reasoning plays the role of abstract instructions, as opposed to instructions specific only to the particular proof and proof step, and increases transferability [Robertson, 2000].

We propose to concentrate on skill acquisition, emphasising implicit knowledge where possible, but not restricting all learning to it. That means that we do not aim at bottom-up concept learning, although the declarative knowledge concerning concepts that the students already possess can be proceduralised during the skill acquisition. We rather aim at bottom-up skill learning. Moreover, we in fact need to rely on some top-down knowledge anyway at a different level, to the extent that we presuppose the lesson material for learning concepts, rules, examples and methods to be used during the tutoring session phase, which simulates implicit learning. This is in line with what Sun and colleagues [Sun *et al.*, 2001] propose.

Since we deal with novices, we do consider instruction that looks at surface characteristics, which is the typical approach they take to problem solving. Such instruction is captured in two aspects of our hinting. First, in the pragmatic aspect that makes use of surface characteristics of the expected answer to guide the student. For example, the number of expected sub-parts in the expected answer. Second, in the use of instructional points that point to surface characteristics of the problem that are present at the point the instruction is provided. We use those as a means to guide the student's attention to the more generalisable underlying characteristics, which lead to procedural knowledge.

The fact that we try to follow what the student does and provide instructions on the proof the student is attempting is an attempt to let students use their knowledge in a way suitable to them, which improves their learning. At the same time, we accept that this use of their knowledge is intractable to both us and them, as it is based on overly complex implicit structures and we do not try to make it overtly available to them by asking them to self-explain. Additionally, we define a teaching model that uses heuristic-based hinting that promotes implicit knowledge, with moderate self-explanation and exposition of declarative knowledge. More specifically, we let students work on their own for as long as they seem to be making progress in the task. Only when this is not working and students seem to slow down, which is an indication of cognitive load, do we interfere and tackle meta-reasoning that provides heuristics declaratively to give students a way out. We, thus, support useful expert schema acquisition, manipulate the synergistic effect, and contribute to the reduction of cognitive load [Stevenson, 1998].

## 2.3 Tutoring Framework and Teaching Model

In this section we formulate the tutoring model that informs the formalisation and implementation of our system. We follow, to a large extent, the phase separation proposed by Collins and Stevens [Collins and Stevens, 1982]. First, we enumerate the assumptions of the tutoring framework that the model presupposes (Section 2.3.1). Second, we specify the tutorial goals, global and local, which the tutoring model sets out to realise (Section 2.3.2). Third, we explain what the specific teaching strategy we implement is, which derives from the theoretical and empirical considerations analysed so far in this chapter, the tutorial goals set, and the general tutoring model (Section 2.3.3).

### 2.3.1 The Tutoring Framework

The tutoring framework presented in this thesis, which we presuppose for the automation of the hinting process in our implementation, comprises the following phases:

1. Lesson material
2. Proof presentation
3. Problem solving session (interactive tutoring session)

*Lesson material* refers to the material that the student consults for preparation. It should include the declarative domain knowledge that the student is supposed to know and that is presupposed for the interactive proving session. *Proof presentation* also belongs to the preparation phase. It should include proofs demonstrating the use of the domain knowledge presented in the corresponding lesson material. Both of these are prerequisites for the final problem solving phase and equally important for the learning process, although they serve different purposes. The *problem solving session* is the interactive session where the student is required to apply the knowledge acquired in the previous two phases, while at the same time practising problem solving skills in general.

The definition and implementation of the lesson material and the proof presentation are not undertaken here, but we assume that they are supplied by the larger system of which this is a module. Hence, we only mention them to the extent that they influence the focus of the thesis, namely the problem solving session, and more specifically the motivation of the hinting process adopted for formalisation. The problem solving session itself is discussed at length in Section 2.2.2. We will, thus, restrain ourselves to saying that the lesson material and the proof presentation should capture the same educational principles (Section 2.3.2.1, global tutorial goal) and the same learning goals of the tutoring session (local tutorial goal) as the problem solving itself. They should be prepared based on a user model of the overall system, which should also determine the appropriate task to be set for the student for each tutoring session. An example of what this consistency should capture is that the proof presentation should be in the form of worked examples that capture the forward proving technique that experts use and also some backward reasoning, where this is necessary [Sweller, 1989] (cf. Section 2.2.2). This would allow students to get a better understanding of the procedure, although the static representation of worked examples is not the best representation for capturing it [Collins and Stevens, 1991]. Then when students are required to find a proof themselves in the interactive phase, they will be more prepared to assimilate and apply dynamic aspects of proving.

Let us now move on to the definition of these tutorial goals.

### 2.3.2 Tutorial Goals

The tutorial goals inform the choice of the teaching model as well as its exact definition and internal choices at all times. For conceptualisation purposes, they can be separated into global and local tutorial goals, although in reality it is hard to tease them apart, as there is a lot of interplay between them.

#### 2.3.2.1 Global Tutorial Goals, Means, and Proposed Strategy

The global cognitive goals that we wish to realise every time through the teaching model are the following:

1. distant transfer
2. near transfer
3. implicit learning
4. self-sufficiency
5. motivation to learn more

*Distant transfer* – also called "far transfer"– refers to the ability of the student to apply any knowledge acquired over a maximally extended period of time, or to problems that do not have surface similarities to the tutored problems. For example, two problems that both require an indirect proof, but do not make use of the same mathematical concepts, should be recognised as similar.

*Near transfer* refers to the ability to use the acquired knowledge under multiple different circumstances and apply it to variants of the tutored problems. For example, when the definition of a concept has been replaced for the instance of a concept in a mathematical tasks, the student should be able to see the same concept in a following similar task and remember to replace the definition for the concept.

*Implicit learning* refers to the acquisition of problem solving skills that are intractable in themselves, but affect the overall performance of students. For example, the student does not need to be aware of the notions of forward and backward reasoning, as most humans aren't, but should be able to apply them, as humans do.

*Self-sufficiency* describes the goal that students reach a point in which they do not need the help of the tutor, and they are confident without it. For example, after solving a number of problems using indirect proofs with the help of instruction by the tutor, the student should be able to recognise that an indirect proof should be applied without any help by the tutor.

*Motivation* to learn more means that the student wishes to indulge in more learning without any external motivation. For example, when students come across difficulties in solving a problem, they should be able to self-regulate their motivation, for instance, by remembering similar situations that they resolved before and by appreciating the exhilaration that resolving difficulties alone can release. They should not need to get external credit or to be reminded that difficulty is part of problem.

#### 2.3.2.2 Local Tutorial Goals

The local tutorial goals that we assume define the learning goals each time, that is, the domain learning goals of each session. In particular we want to teach:

1. Application of domain concepts
2. Application of new theorems/lemmata/definitions/axioms
3. Application of new proving techniques (controlled through the choice of proving task)
4. Correct use of argumentation (deductive reasoning)
5. Correct notation
6. Correct jargon

These goals are realised in every proving task keeping in mind the global tutorial goal that is captured by our tutoring model. The latter is informed by state-of-the-art studies in educational and cognitive psychology (learning theories) as examined in this chapter.

### 2.3.3 The Teaching Model

We propose the simulation of a non-goal-specific instructional teaching model. More specifically, we adopt the problem solving paradigm, which is more appropriate for our tutorial manager, as this is a module responsible for the interactive phase of the learning framework. We use non-goal-specific problem solving, which advances the benefits of problem solving in the training phase and lessens the problem of extra cognitive load imposed by the goal-oriented strategies. For the domain of proving this translates into allowing and supporting students to find any solution to the task. Moreover, we choose instructional problem solving and with it the *`Socratic`* teaching model, driven by the goal to further reduce any unnecessary cognitive load, take motivational issues into account, and allow for more fine-grained manipulation of the tutoring session towards our global tutorial goals. In that context, we also allow for making use of adaptive natural language realisations of the instructions provided. Furthermore, we take into account the divergent views on promoting declarative (explicit) or procedural (implicit) learning and decide on a synergistic model with emphasis on implicit learning for our teaching model. This means that we aim at implicit learning, with minimal explicit learning only when judged necessary.

In addition, as we have seen (Section 2.2.1), schema theory underlies our teaching model and all decisions relevant to it. Schema theory and cognitive load theory are adapted for teaching problem solving by instructional learning theories [Rumelhart and Ortony, 1977; Rumelhart, 1980; Price *et al.*, 1997; Paas *et al.*, 2004; Widmayer, URL]. We borrow the ideas from these three theories. We use the schema analogy and aim at the active building of logical expressions, which are the building blocks of schemata, at an abstract level. We use instructional points to represent the building blocks of schemata and to provide attentional anchors for procedural skill acquisition. For that purpose, we abide by instructional theory that advocates in general the teaching of mental models for schema acquisition and stresses the need for multiple exposure to schema building situations before the schema can be re-applied fluently. This is reinforced by appropriate instruction (Section 2.2.5). In the same spirit, the common criticism against the claim that learners must acquire problem solving abilities through exposure to known situations, points out that schemata acquired this way may over-fit the learning situations, become rigid, and useless for more general

application [Wu, 2001; Widmayer, URL]. A case of over-fitting the data for the schema example in Section 2.2.1 would be that the learner acquires a schema for solving equations of the form (a + b)/c = d with natural numbers only in place of the variables a, b, c, d and cannot use the same schema for the same sort of equation with real numbers. Additionally, there is evidence that problem solving is domain specific in nature, and students need to learn through domain-specific knowldge so that they can approximate the mental representations of domain experts [Chi *et al.*, 1982; Schoenfeld and Herrmann, 1982; Sweller, 1989]. These two views taken together suggest that teaching abstract mathematical principles applied to specific learning situations and domain knowledge as the best known way to acquire problem solving

skills [Driscoll, 1994; Price *et al.*, 1997; Widmayer, URL]. Therefore, we employ such principles as they also offer themselves for instructing the students to a useful way of problem solving for the particular theory taught [VanLehn *et al.*, 2005]. We also implement a hinting style which guides the student to the more efficient directionality, forward or backward, at each point, provided that motivational issues are warranted. We base the hinting line on heuristics that constitute a proving path, equivalent to a possible schema, and point the students to a possible proof without imposing it as the only solution (See Chapters 5, 4 and 6).

#### 2.3.3.1 Characterisation of the Tutoring Model

We now provide a characterisation of our tutoring model based on the established work by Collins and colleagues [Stevens and Collins, 1977; Collins and Stevens, 1982; Stevens *et al.*, 1982; Collins and Stevens, 1991; Collins, 1991] for reference purposes. We make use of the issues for tutoring identified by Wilson and Cole [Wilson and Cole, 1991] and largely taken over from Collins [Collins, 1991]. We briefly position our model with regard to each of these categories.

**Content** In terms of content taught, we provide domain knowledge typically found in textbooks when it is relevant to the proof at hand, but it is not our goal to teach this knowledge independently of finding the proof. We also employ heuristically based instruction (cf. Section 2.3.4), but we do not enforce heuristic strategies or aim at metacognitive skills as such, where students are required to explicitly learn to monitor their own progress.

**Situated learning** Theorem proving is a real-life application of problem solving. We emphasise problem solving as part of our teaching strategy and aim at learning through experience, since this is also a prerequisite of schema acquisition. The combination of those two aspects makes our approach affiliated to situated learning, in the wider sense [Collins, 1991]. However, we provide hints and additional instruction to accelerate and enhance learning through experience (cf. Section 2.3.4), which is not part of standard situated learning.

**Modelling and Explanation** Modelling of the proving process is inherent in our domain to the extent that we allow students to find a proof dynamically and, hence, appreciate the dynamic evolution of proofs, as opposed to the static representation included in textbooks. For that reason, we also partially model the expert performance by supporting forward steps, but allow the student to take potentially more backward steps than an expert might do. We do not work with explanation of worked examples, though, as this is not part of the particular phase in tutoring that we are concerned with here.

**Coaching** Coaching is a main characteristic of our teaching strategy. Issues of cognitive load, motivation, and schema acquisition are all pointers to recognising which information, how much information, and in which way information should be presented when coaching the student. Our teaching strategy is based on getting the students to do as much as possible on their own and intervene only when coaching is necessary. However, we do not view problem solving just as a way to create the teaching moment [Wilson and Cole, 1991], but we rather see problem solving itself as the platform for learning through experience, while coaching is only a corrective mechanism for this kind of learning.

**Articulation** We aim at minimal articulation and explicit learning at points where implicit learning seems to fail.

**Exploration** Our model includes some exploration because we encourage students to work towards a solution of their own in real-life application such as theorem proving. We let them take back their current contribution or the step they are working on and try another one, or to start a completely new proof. Still, we do not promote exploration in the sense that students set tasks for themselves. The system sets the tasks that students work on.

**Reflection** Reflection is promoted via hints in our model in so far as we make use of the students' own reasoning and provide feedback based on that. This forces students to reflect upon their own answers [Tsovaltzi and Fiedler, 2003a].

**Sequencing** We do not deal with the choice of tasks that is essential to Collins and colleagues [Stevens *et al.*, 1982; Collins and Stevens, 1982; 1991]. Therefore, sequencing from less to more complex tasks falls out of our scope of research. Our aim is to provide feedback on proofs and their proof steps after the proof task has been chosen. We guide the student on performing the next proof step and provide feedback on errors during the performance of a step. This may involve subtasks where a misconception is remedied, but it does not involve presenting the student with a new task that will help resolve the misconception.

On the whole, as our aim is to automate feedback, we rather take an approach of looking for the defining characteristics of every tactic proposed by Collins and colleagues. Namely, we identify the underlying psychological or educational principle that makes the tactic applicable. These inform the multiple decisions that the tutor makes at every point and are implemented in our model as the choice of general pedagogical feedback, choices on the multiple hint dimension (Chapter 4), and choice of substrategies (Chapter 6) that all contribute to the final decision on the appropriate feedback.

### 2.3.4 Guidelines for the Realisation of our Tutorial Goals

Let us now briefly explain the methodology for realising these goals in a concrete teaching strategy and its implementation, as instructed by the research reviewed in this chapter.

We undertake a full automation of hints, i.e., we employ the concept of proof step matching (cf. Section 2.2), and regulate our feedback according to it in order to follow the students' own line of reasoning. *Proof step matching* means that the student's attempt, underspecified as it might be, is matched to one of the possible correct proof steps, which we call the *expected proof step*. Then the subparts of the attempted proof step are also matched to the expected-step subparts[4]. The result of this comparison is an evaluation of the student's attempt, which depends on domain knowledge and constitutes input to `Menon`. We also use tutoring concepts – the instructional points – defined as part of a domain ontology, which allows matching specific mathematical concepts, relations and terms in the proof step, to the abstractly defined instructional points (cf. Chapter 3). The extraction of hint categories is based on these instructional points and the instantiation of the domain-specific knowledge of hints is facilitated by the domain ontology. This way, `Menon` produces feedback for the proof steps that the student chooses to attempt on the fly and there is no need to impose a precompiled solution. Moreover, our hinting strategy provides help primarily when the system (tutor) judges that help is needed, and not when the student asks for it, as evidence supports that the latter may be a weak tactic as students are generally bad in monitoring their own progress and need for help [Aleven and Koedinger, 2000b].

Motivation comes into play in our teaching model both in the form of explicit encouragement captured verbally, as well as in the form of informed tutoring choices, which aim at promoting attention, relevance, confidence and satisfaction (cf. Section 2.2.4). The personalisation of the learning process is an additional motivation technique in its own right [Ross and Fulton, 1994].

Reduction of cognitive load relates in a seemingly antagonistic way to motivation. The latter supports challenging the student, which translates into an effort requirement, while cognitive load theory emphasises the importance of providing right amounts and the appropriate level of help to alleviate unnecessary effort. This aim is undertaken, as much as possible, in the definition of the hint taxonomy (cf. Chapter 4), which identifies small meaningful domain-knowledge chunks along the lines of schema theory, and in the hint-choice strategy, which evokes these knowledge chunks for instruction based on the performance of each student (cf. Chapters 4 and 6). The student's performance is assessed by help of a model that we call the *Hinting Session Status* and represents the status of the session (cf. Chapter 5). Hence, our goal is that the organisation of the instructions relevant to the schema acquisition is responsible for any necessary cognitive load imposed [Paas *et al.*, 2004].

To promote the forward-proving technique typically applied by experts [Chi

[4] In the Dialog project, a proof manager based on the theorem prover Omega [Siekmann *et al.*, 2002] was used to this end. For more details see Chapter 5.

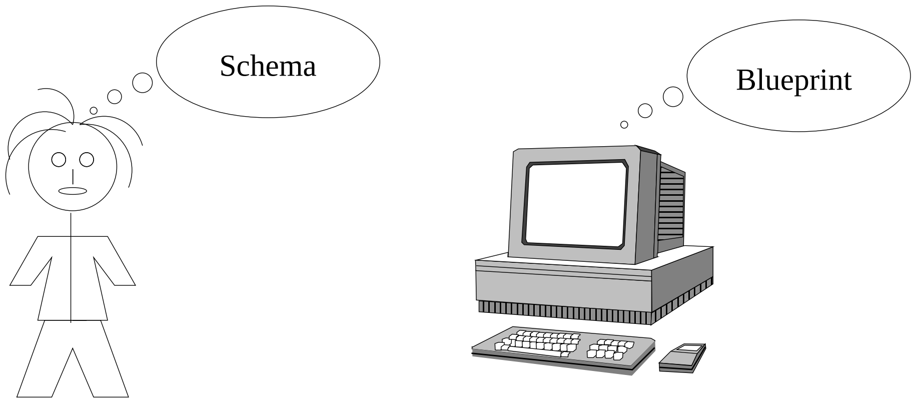


Figure 2.1: There is no one-to-one correspondence between the instructional blueprint and the schema that it helps give rise to.

*et al.*, 1982; Sweller, 1989], our hint-choice strategy helps students to make forward steps whenever possible. Thus, we also encourage them to make use of deep domain structures that requires the application of a schema [Sweller, 1989; Koedinger and Anderson, 1990]. The alternative, that is backward steps characteristic of novices, makes use of surface structures, such as what the required goal looks like. On the other hand, our strategy helps students when a backward step is on call, hence allowing them to consider the goal when this is necessary, but preventing that the students move aimlessly back and forth. This means-end technique [Koedinger and Anderson, 1990] is necessary for novices to acquire schemata, even though it becomes obsolete when the schema already exists.

The aim of assisting the construction of cognitive structures (mental schemata) directs the way we organise our domain ontology, as well as the definition of our instruction units (the hints) and our hint-choice strategy. We do not make any ontological claims about schemata. We only borrow the state-of-the-art theoretical construct schema from cognitive psychology to explain learning. In the rest of this thesis, the mention of schemata is used only with that intention, while we define a blueprint to use as our hint-choice strategy, which is a possible heuristic model for problem solving in our domain, as suggested by instructional theory (cf. Figure 2.1). The definition of the blueprint is a two-step process. First, we define instructional points for tutoring problem solving based on the pedagogical considerations that we have motivated, on analysis of empirical data [Moore, 2000; Benzmüller *et al.*, 2003b; Wolska *et al.*, 2004] (see also Chapter 1), and on the structure of our domain [Schreiner, 2004]. Second, we make use of these instructional points and how they relate to each other for the derivation of the blueprint. The definitions of the blueprint and the instructional points can be found in Chapter 3.

Implicit learning is another consideration for the way our teaching strategy is organised. We follow Mathews and colleagues [Mathews *et al.*, 1989], who propose a framework in which implicit learning is the mechanism for recognising family resemblance. This recognition is essential for the application of the appropriate schema. Therefore, the use of the general blueprint to guide students to acquire a potential schema of their own aims at such implicit learning. The schema acquired cannot be identified exactly and is also not our aim to observe it.

In general, we subscribe to instructional theory and, through it, to the application of schema theory to tutoring [Delclos and Harrington, 1991; Driscoll, 1994]. Following Lim and Moore [Lim and Moore, 2002] as well as Cooper and Sweller [Cooper *et al.*, 1999], we suggest the use of a blueprint to represent a generic pattern of moves towards solutions in mathematical proofs that capture the underlying similar solution path for seemingly different problems. Our purpose is not to impose that blueprint on the student. Rather we want to use the blueprint as a guide to the problem solving in our domain and the way the instructional points can be presented to the student, as Stanley and colleagues [Stanley *et al.*, 1989] propose. Both the instructional points and the blueprint definition follow this general philosophy.

A more detailed presentation of the actual Socratic strategy that we implement, expanding on how the teaching guidelines mentioned in this section are realised, is presented in Chapter 6.

## 2.4 Conclusion

Our approach is oriented towards modelling the student's attempt with respect to the task in the domain. We break the proof steps into their constituent domain knowledge - the instructional points - and shift the evaluation of correctness to those constituents. Therefore, a proof step as a whole is attributed a third correctness value, apart from correct and wrong, namely partially correct, when only some of the constituents are correct. In modelling the student's attempts, we track these instructional points and provide feedback accordingly. We count on the student for building the schema required for learning, in that we only base our instruction on these instructional points and on an abstract minimalistic heuristic blueprint abiding by Wilson's statement that "Given the right design, incomplete learning specifications can nonetheless lead toward complete learning outcomes" [Wilson and Cole, 1991] (p.12). This has two advantages. First, that it supports our intention to not superimpose any assumed best cognitive structure, but try to help students to learn on their existing structures. Second, that it makes the cost of implementing automatic feedback manageable (cf. Chapter 1).

The rest of this thesis is concerned with the necessary definitions and use of artificial intelligence techniques for building a system, `Menon`, which implements the derived teaching model. The approach to producing automated feedback for arbitrary problems and solution, and the implemented teaching strategy are described.

# Chapter 3

# Instructional Points and Blueprint for Problem Solving

## 3.1 Introduction

In Chapter 2, we argued for the importance of the underlying domain principles in tutoring and we motivated the definition of instructional points and of an instructional blueprint that represents a possible reasoning for arriving at proof steps. In this chapter, we look more closely into the instructional points and we define our instructional points and blueprint as part of an ontology for adaptive hinting. In this context, we also present the proving domain and the OMEGA system as a suitable domain reasoner for tutoring in this domain.

The role of the ontology derived in this chapter in automating hinting is twofold. First, it influences the choice of the appropriate hint category since it is represented in the Hinting Session Status (HSS) that is the input to the Socratic tutoring strategy [Fiedler and Tsovaltzi, 2003a]. Second, it determines the domain content of the hint to be generated. More specifically, it can be used to map the instructional points, which are the domain-information specifications of hints, onto their instantiations in the proof and proof step at hand represented in the domain reasoner. The ontology was derived by combining a top-down and a bottom-up approach. The top-down approach was the theoretical exposition of schema acquisition, which provided us with the notion of instructional points and blueprint for structuring the domain for tutoring and was presented in Chapter 2. The bottom-up approach consisted of two methods of analysis: (i) The exploration of the domain for potentially useful concepts and relations to consider as instances for our instructional points, which was done in preparation of the Wizard-of-Oz experiment of the DIALOG project [Tsovaltzi and Fiedler, 2003b]. (ii) The data analysis of the Wizard-of-Oz experiment, where we aimed to identify those useful instructional points in students' common mistakes.

We first define abstract instructional points for tutoring and the blueprint for problem solving that makes use of these instructional points in Section 3.2. To this end, we try to stay faithful to previous definitions of schemata and empirical evidence on schema acquisition. We then introduce the formalisation of the abstract instructional points as part of the domain ontology for tutoring proving in Section 3.4. We base our formalisations on our bottom-up analyses.

## 3.2 Definition of the Instructional Points and the Blueprint

For the derivation of the instructional points we follow the Eysenck and Keane's [Eysenck and Keane, 2000] definition of schemata. According to this, schemata consist of relations, variables, and values for the variables. Relations can vary from binary to causal. Variables are place holders for concepts or other schemata that have to fulfil certain pre-conditions. It follows that schemata can be embedded in other more general schemata. A situation is considered an instance of a schema, although not all possible variables are already instantiated. In that case, the corresponding relations of these variables cannot be evaluated either.

Domain knowledge comprises *anchoring points*, which are elementary for the acquisition of the schema by the student and all other knowledge is based on them. In our domain, such anchoring points are meaningful subparts of the proof, which serve as building blocks for the whole proof. A new situation might fit the schema representation by matching some or all of these anchoring points, hence, making up an *instance* of the schema [Russell and Norvig, 2003].

Anchoring points are used to describe the organisation of knowledge in the human mind. Since our aim is to assist the creation of this knowledge and not to describe it, we are interested in defining corresponding instructional points for teaching purposes. Therefore, we talk about *instructional points*, a name reminiscent of the instructional theory paradigm that it echoes. Our assumption is that by using the blueprint with its instructional points to instruct the student via hints, the possibility is promoted that the student acquires a related schema. This will help students learn in a structured way and assist in improving their skills in the task.

We define instructional points that are meaningful domain-knowledge units, following research in the field of genetic algorithms that demonstrated that using an analogous representation of schemata as comprising of meaningful subparts works best. As an example outside the proving domain, meaningful subparts for representing an antenna are reflectors, deflectors etc. [Russell and Norvig, 2003]. In our domain straightforward equivalent examples are the proof steps of a proof and the rule used as the justification for making a specific proof step.

In general, we identify appropriate instructional points informed by experimental evidence on the underlying deep understanding of experts, as opposed to surface characteristics that novices depend on for solving problems [Schoenfeld and Herrmann, 1982; Chi *et al.*, 1982]. The same studies also suggested that this deep understanding is required for solving problems. The underlying principles of mathematics that experts are guided by and that we wish students to acquire as knowledge should be captured in our instructional points [Owen and Sweller, 1985; Sweller, 1989; Lim and Moore, 2002; VanLehn *et al.*, 2005]. We borrow this structure as depicted by Wolfgang Schreiner for formal proving [Schreiner, 2004] and modify it to make explicit further implicit points in the structure that are useful for tutoring. Still, we do not want to involve students in the formal proof techniques directly, which may be unintuitive for humans, but want to instruct students in the standard way proving is communicated among humans, that is using the mathematical vernacular.

On the other hand, our job is to provide a way of uncovering these underlying similarities between problems, which are obvious to experts but not to novices. Chi et al. [Chi *et al.*, 1982] have found that experts and novices can recognise the relevant information alike. The difference in their expertise lies on the reasons for perceiving them as relevant, or on the knowledge that relevant features trigger. In particular, to identify relevant features, they both have to rely on surface characteristics of the problem. However, novices rely on these surface characteristics for classifying the problem. Experts, on the other hand interpret them as pointers to the underlying principles and categorise problems based on those underlying principles. Therefore, we include instructional points that correspond to the feature characteristics, but also ones that reveal their use as pointers to the underlying principles. We associate the second with meta-reasoning.

We now define our abstract instructional points for a proof and a proof step. *Proof steps* are solution steps that comprise applications of domain rules to previous proof steps, making up the chain of reasoning, which is the proof itself. We refer to such domain rules in our domain as inference rules and to proof steps as inference steps. The following is the standard form of an inference step in set theory, where $\varphi_1, \ldots, \varphi_n$ are the premises, $\psi$ is the conclusion, and P is the inference rule:

$$\frac{\varphi_1, \ldots, \varphi_n}{\psi}\mathrm{P}$$

We categorise the instructional points in classes to capture the structure of the domain. We shall then discuss the different classes.

Let $E_s$ be the expected proof step in a particular proof, consisting of the premises, the conclusion and the inference rule applied. Let $P_i$ be an instructional point. We say that $P_i$ is an instructional point for $E_s$ if $P_i$ falls under one of the following categories:

**Domain Relation:** $P_i$ is a binary relation between mathematical concepts in the domain. For example, antithesis holds between $\in$ and $\notin$. These relations hold independent of $E_s$.

**Domain Object:**

*Domain Object:* $P_i$ is a function representing a relation between the premise/-s, the conclusion, the inference rule (see below) and an entity in the proof step that constitutes a prerequisite for the application of the inference rule for making the inference in $E_s$. In Section 3.4 we define Relevant and Subordinate Concept, as instances of Domain Object.

*Relevant-Concept Meta-Reasoning:* $P_i$ is a relation between the Relevant Concept and $E_s$. That is, considering the Relevant Concept helps making the inference in $E_s$.

*Subordinate-Concept Meta-Reasoning* $P_i$ is a relation between the Relevant Concept the Subordinate Concept and $E_s$. That is, considering the Relevant Concept in connection to the Subordinate Concept helps making the inference in $E_s$.

**Rule of Inference:**

*Rule of Inference:* $P_i$ is the rule used as the justifications for making the inference in $E_s$ from the givens.

*Domain Technique:* $P_i$ is a relation between the Rule of Inference and the Relevant Concept in $E_s$. For example, case split is the Domain Technique in $E_s$, where the Rule of Inference are the different cases deriving from a disjunctively defined concept.

*Connect Relevant-Subordinate-Concept:* $P_i$ is a relation between the domain objects and the Rule of Inference. Namely, that the applicable Rule of Inference involves the domain objects; Relevant and Subordinate Concept.

*Elaborate Domain Object* $P_i$ is a relation between the Relevant Concept, the Rule of Inference and the Domain Technique in $E_s$, namely that the Domain Technique is used to apply the Rule of Inference with regard to the Relevant Concept.

**Substitution:**

*Substitution:* $P_i$ is a relation between the premise, the Rule of Inference and the conclusion, which stands for the application of the Rule of Inference for $E_s$.

*Inference-Rule Application:* $P_i$ is a relation between the Domain Technique the Rule of Inference and the relevant concept. $P_i$ represents the substitution of the relevant concept based on the Rule of Inference.

**Proof step:**

*Starting Point:* $P_i$ is as a relation between the premise, the conclusion and $E_s$, representing that the reasoning about the step should start by identifying the premise and the conclusion.

*Premise-Conclusion:* $P_i$ are the premise/-s and conclusion of the present expression.

*Abstract Method:* $P_i$ is the abstract method applied in the proof. We consider direct and indirect proof as abstract methods.

*Specific Method:* $P_i$ is the specific method applied in the proof. We consider forward and backward steps as specific methods.

*Step Meta-Reasoning:* $P_i$ is a relation representing the connection among the different parts constituting the proof step. It includes, for instance, meta-reasoning-relevant-concept and meta-reasoning-subordinate-concept.

Our instructional points are ordered with respect to the amount of information they reveal towards the completion of a proof step. We call this ordering relation *subordination* and it captures the partial ordering that human tutors use as a strategy for choosing their next tutoring action. It serves as an indication of whether the student might or might not know a domain-knowledge element. If the student knows an element that presupposed the knowledge of other elements, it can be safely assumed that the student also knows those elements [Collins and Stevens, 1982]. Exceptions to this rule can derive from partial ordering of inference steps. Other exceptions might be hidden misconceptions. However, this is a complicated issue, which requires special treatment (cf. Chapter 6). The subordination relation helps us derive the gross blueprint for problem solving, and it also takes care of motivation issues discussed in Section 2.2.4; For example, providing the right amount of information at each time.

The `domain relation` class captures relations between concepts in the domain. A better understanding of such relations allows noticing patterns for schema acquisition and selection [Price *et al.*, 1997; Koedinger and Anderson, 1990].

The `domain object` instructional points capture the fact that selection of schemata is based on relevant information on which attention should be concentrated [Collins and Stevens, 1991; Price *et al.*, 1997]. The meta-reasoning instructional points, explain this reasoning with respect to the different domain objects defined. In schema theory terminology [Rumelhart, 1980], domain objects and their meta-reasoning are the equivalent of "feature detectors" (p. 42).

The instructional points of the class `rule of inference` straightforwardly capture the fact that there must be a rule of inference. Rules of inference are organised in more general categories of domain techniques. When a domain technique applies, only the rules of inference which make use of it may be applied. The domain technique is useful for reducing the search space of rules of inference that apply in a situation. The meta-reasoning instructional points explain how the domain objects can help to derive the Rule of Inference and the domain technique. In terms of schemata, this relation captures the way variable binding is constraint by the presence of already bound variables [Rumelhart, 1980].

The `substitution` class instructional points refer to the final use of the previous instructional points for actually deriving the proof step.

The instructional points of the class `proof step` deal with relevant information to be considered for the proof as a whole and the relation of the proof step to the proof. They also explain how this information can be used to derive the next step. Abstract Method has been specifically shown to be beneficial for problem solving tasks [Price *et al.*, 1997].

The instructional points in the classes `domain relation`, `domain step`, `rule of inference` and `proof object` suggest a way of understanding [Greeno, 1978; Chi *et al.*, 1982] the problem. This means that they are the elements of the representation of the initial state of the problem and the goal (Starting Point and Premise-Conclusion in `proof object`), or of the problem-solving operators (`domain objects`, `rule of inference`), and of the different relational structures among them (`domain relation`). The class `substitution` corresponds to the inferences that can be made based on the above elements.

Instructional points are also separated into two categories: performable-step instructional points, which are directly relevant to deriving the proof step, and meta-reasoning instructional points, which are relevant to instantiating the former. Performable-step instructional points are defined in the classes `domain object`, `rule of inference`, and `substitution`. The rest of the classes include only meta-reasoning instructional points.

Let us now turn to the blueprint itself. In general, the *blueprint* represents a possible reasoning for finding proof steps. Instructional points are the building blocks of our domain rules, which need to be mastered. Hints either point to instructional points and their connection to the domain rules, or they point to the rules directly. Alternatively, they point to meta-rules for deriving a schema, which in effect make up the blueprint itself (cf. Chapter 4).

Based now on our instructional points and the considerations mentioned above, we define the problem solving blueprint for deriving a proof step, represented as a graph in Figure 3.1[1]. The nodes represent instructional points. `Step meta-reasoning` is not included as it only concerns tutoring and not the decision process towards a step itself. Some nodes include both the performable-step instructional point and the meta-reasoning instructional points relevant to it. We capture the loop of trying to instantiate the performable-step instructional point this way for the sake of simplification and clarity. The directed arcs represent decisions on which point to consider next. “No” loops represent the places where the instructional point in question should be directly addressed next, so that if the premise and the conclusion are not known, they should be addressed next. The blueprint guides the instructions generated by `Menon` as a means to help the student out of endless looping. The specific instructions are investigated in Chapter 6.

Our blueprint attempts to represent the most abstract possible schema, bringing together similarities and common solution steps among all problems

[1]Note that the blueprint does not represent error debugging, which is dealt with in Chapter 6.

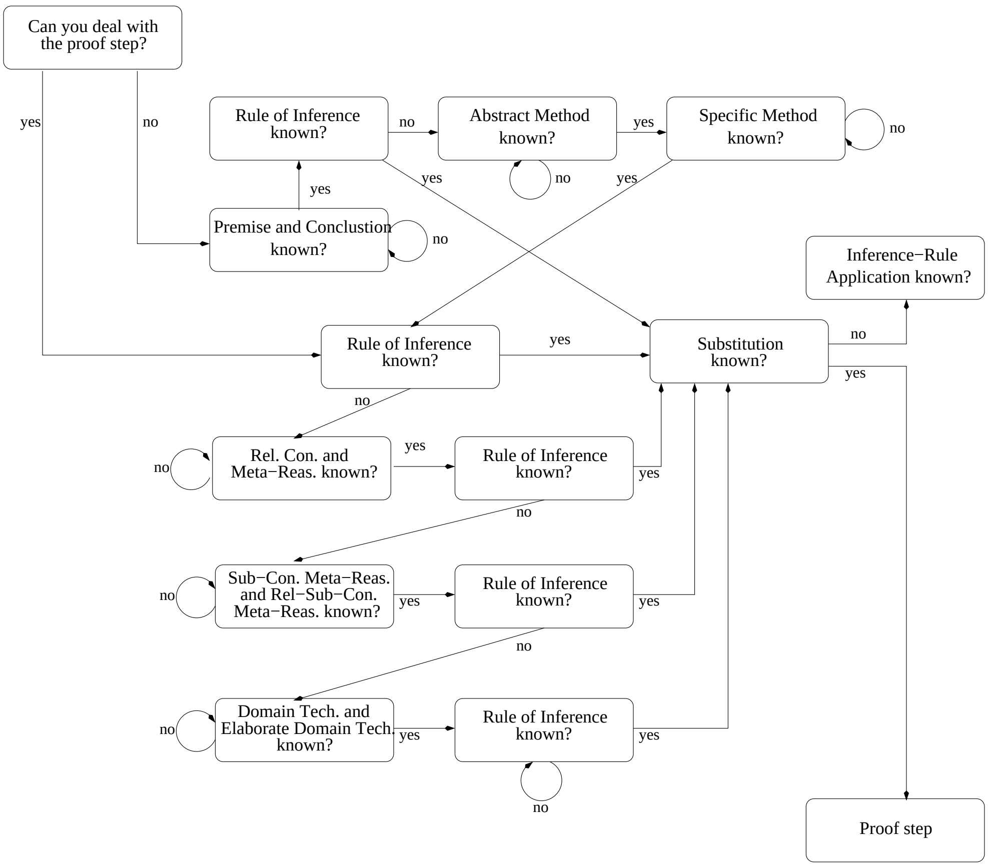


Figure 3.1: The instructional blueprint as a graph.

in the domain [Sweller, 1989]. Moreover, the blueprint, following the subordination relation, makes use of the idea that the basic sub-schemata activate the related more abstract super schemata, which then can activate more sub-schemata [Rumelhart, 1980]. The activation is consequently spread across all relevant schemata. Therefore, domain relation and domain objects that are more elementary are addressed before the Rule of Inference, which is more abstract and depends on the domain objects (cf. Section 3.4). The Substitution is addressed after the Rule of Inference as it is even more specific.

Proof step instructional points especially represent the most abstract domain principles. Chi et al. [Chi *et al.*, 1982] identify underlying domain principles used by experts, including the justifications applied, with methodological principles. Following this and the other theoretical standpoints summarised in this section, and especially the work by Koedinger and Anderson [Koedinger and Anderson, 1990], the blueprint starts by addressing proof step instructional points, as these are common among many proofs. For instance, since proofs can either be direct or indirect in terms of the Abstract Method, many proofs will share the same Abstract Method. This already narrows down the search space for the schema. However, the blueprint points to the proof step as a whole only if the student does not seem to have the information needed (represented by the instructional points). Therefore, the blueprint only requests that information about the proof method is explicitly handled if no schema is available which would allow the student to reason about the proof method implicitly. If students still cannot proceed alone, then they are pointed to how to reason about finding the right inference; First abstractly, and then via the use of what domain knowledge is available in the expression.

In Section 1.3.4, we saw a possible path through this blueprint created interactively by the student and the tutor. That path was as follows. The student knows how to deal with the proof step (**S1**, Starting Point), so the tutor elicits the premise and the conclusion (**T2**, premise-conclusion). The student finds the premise and the conclusion (**S2**), but does not know the Rule of Inference, so the check for the rest proof-step meta-reasoning instructional points is done. Since the Abstract Method is direct, it is considered known, but the student does not know the Specific Method, which is elicited (**T3**), and the student provides it (**S3**). The Rule of Inference is still not known, so the search for the domain concepts starts and the Relevant-Concept Meta-Reasoning is addressed, as a result (**T4**). The student finds it (**S4**), so it is known for the next cycle through the blueprint, and the next instructional point addressed is the Subordinate Concept (**T5**), as neither the Rule of Inference is known yet, nor is there any concept used that bears a domain relation to the Subordinate Concept. When the student provides the Subordinate Concept (**S5**), the Rule of Inference is addressed (**T6**). As soon as this is known (**S6**), the Substitution is addressed (**T7**). The student finds it (**S7**) and the tutor prompts for the next step (**T8**, Proof Step), which closes a complete iteration. The implementation of the blueprint in our Socratic strategy is explored in Chapter 6. More possible paths through the blueprint are also detailed in the examples in Section 6.6.


Figure 3.2: Overview of the domain ontology

## 3.3 Mathematical Domain Knowledge: The OMEGA System

This thesis proposes a general approach to automating feedback in tutorial dialogues. As an application of this approach, we explore the domain of proving in set theory. In a tutoring system for proving there are two aspects of representing the proof. The first aspect refers to the proof which we want to teach the student. This aspect addresses the needs of tutoring proofs at the level found in a textbook. The second aspect considers the system-level of proofs which is necessary for the automated reasoning inside a domain reasoner and can provide the information necessary for tutoring a proof. `Menon` represents the tutored-proof level through instructional points for tutoring which mediate the proof representation for tutoring and the system-level of proof represented in a domain reasoner. In the architecture that we discussed in Section 1.3.2, the domain reasoner responsible for the system-level proof representation is the OMEGA system (see Figure 1.1). Although this representation needs to be extended for tutoring proofs, the OMEGA system represents human-oriented proofs which is far closer to the tutored task than the standard logic-level proof. We will now describe the OMEGA system and the proof representation in it in more detail.

OMEGA [Autexier *et al.*, 2008; Benzmüller *et al.*, 1997; Fiedler *et al.*, 2002] is a mathematical assistant system which aims to support working mathematicians in their everyday research. OMEGA uses various proof search techniques which all work on a main *proof object*. This is represented in the *proof data structure* (PDS) which is a collection of directed acyclic graphs making up a proof forest. The PDS is an elaborate representation of a proof at different levels of abstraction from more detailed (logic level) to less detailed (human-oriented level). It also records alternative proof attempts. This representation is dynamic as it maintains the status of the proof search during the development of the proof. Each conjectured *lemma* is represented by its own tree and it becomes a task to be proved by reducing a *goal* to a conjunction of subgoals. Each lemma can be applied to each proof tree as an inference. Different reductions of the same goal give rise to different proofs, or different abstraction levels. Subgoals comprise tasks and subtasks for the *TaskLayer*.

The TaskLayer is the central component in the OMEGA architecture where tasks become concrete proof-construction steps. It supports representation at the *assertion level*, meaning that definitions, axioms, theorems and hypotheses can also be used as justifications of proof steps, much like in textbook proofs. Assertion-level theorem proving thus moves away from the machine-like logic-rule justifications. The TaskLayer also supports defining foci of attention on subformulas. A task is reduced to a possibly empty set of subtasks using proof construction steps like weakening and decomposing subformulas, or the application of *inferences*. Inferences are the basic reasoning steps and may take the form of assertion application, proof planning methods, or calls to external specialised systems (e.g computer algebra systems). Each inference is a proof step consisting of premises and conclusions. Additional information on such proof steps includes:

- possible hypotheses for each premise (in the case of backward applications of inference rules)
- *application conditions* for the inference as a whole,
- *completion functions* that compute the values of premises and conclusions from other premises and conclusions and
- an *expansion function* that refines the abstract inference step

Omega supports proof planning at an abstract level where an incomplete proof sketch is created, rather than the expanded proof[2]. The basic unit of a proof plan are *methods*, that is plan operators that represent mathematical techniques applied by mathematicians. *Control rules* are used to decide between various applicable methods. An ordered accumulation of methods and control rules is called a *strategy*. Strategies are particularly useful in capturing the tricks of the trade for domain-specific problems.

A reactive parallel proof search is applied based on the Ω-Ants-system. This system supports capturing each inference into an agent, the *inference ant*. Inference ants inspect the proof development and bid for their application at a proof state, resulting in potentially multiple bids in every proof situation. The task of the Ω-Ants-system is to find complete partial argument instantiations, in which case the inference is considered applicable. This architecture allows proof construction by joining the forces of various heterogenous reasoning systems.

Knowledge representation in Omega as a whole is organised as a hierarchy of nested mathematical theories. Each theory includes definitions of mathematical concepts, lemmata and theorems about them, and inference rules, which can be seen as lemmata that the proof planner can directly apply. Moreover, each theory inherits all definitions, lemmata and theorems as well as all inference rules from higher theories.

As mentioned above, the most important aspect of the Omega system for our work is that, whereas classical theorem provers represent proofs at the logic level of particular proof calculi, which is far from the textbook proof that we aim to tutor, the abstract-proof representation in the Omega system makes it an optimal candidate as a domain reasoner for tutoring proving. In the context of tutoring mathematical proofs like the one investigated in the Dialog project, a domain reasoner is expected to dynamically process the informal student's input for soundness, relevance, granularity (level of detail), and ambiguity resolution which is caused by the informal and underspecified student input. Omega was used (i) to reconstruct proof steps from such informal and underspecified student attempts, (ii) to represent and maintain the mathematical knowledge the student is allowed to use and (iii) to maintain the open-ended developing proof and the varied proofs that may derive from it.

[2]Note that this approach is consistent with cognitive models of expert-like problem solving [Koedinger and Anderson, 1990]

All these points together are relevant to evaluating the student's attempt with respect to correctness and completeness. Completeness here refers to the human-oriented proofs where a proof may be considered complete although a lot of information represented in a formal proof is omitted. Especially maintaining the open-ended proof attempted by the student, which is done in the PDS, is particularly important in the context of this thesis as it is a prerequisite for providing feedback in the form of hints to help the student perform a next appropriate step. In general, OMEGA calculates the possible development of the proof after each student attempt, which is a proof state in the PDS. If no possible proof development can be found from the student's attempt, then this attempt is incorrect. If one or more possible proof developments are found from the previous attempt and a new student attempt matches one of them, then the new attempt is correct. If the student cannot proceed and does not enter any new attempt, the possible developments calculated for the previous student attempt, can be used to lead the student to a complete proof.

In the OMEGA group, a language of underspecificaton (S0) has been created to extend the proof represantation for tutorial dialogues [Autexier and Fiedler, 2006]. S0 is an example of how the values for the instructional points can be derived for the NL generation of hints. By use of such a representation language, the proof manager can compare the expected (correct) answer to the student input attempted and thus provide: (i) information on existing mistakes in the student's attempt, which the student may need to correct, (ii) the missing information from the student's attempted step, which the student may need to complete, (iii) the information on the reasoning for achieving the missing parts of the next step, or the whole step, which can be used appropriately for guiding the student. These are the three categories of information required by our hint specifications (See Chapter 4). The possible proof development represented in the PDS is the source of the domain information for instantiating our instructional points.

### 3.3.1 Enhancing the OMEGA Ontology

Reasoning about the completeness of a proof and proof step for tutoring purposes requires defining parts of them that are essential even for human-oriented and textbook proofs. An additional parameter relevant to completeness in a tutoring setting are the pedagogical goals that the tutoring is trying to achieve. This parameter restricts the range and defines the kind of domain information that is relevant to tutoring and needs to be tracked, identified, and evaluated for correctness by the domain reasoner, or has to be provided by the domain reasoner for use in hinting. In our approach, instructional points define such domain knowledge and hence mediate the task of a domain reasoner such as OMEGA in a tutoring setting.

In view of the Wizard-of-Oz experiment, conducted in the DIALOG project, and in order to capture the domain needs for hinting in tutoring and define instructional points, the existing domain ontology for set theory of the OMEGA system [The OMEGA group, url] was enhanced. Since definitions, lemmata, theorems and inference rules in Omega draw on mathematical concepts defined in the mathematical theories, the mathematical database implicitly represents many relations that can potentially be used in tutorial dialogues. Those implicit relations can inform the attempt to identify the instructional points that need to be addressed, and can capture the kind of student reasoning that learning should be based on, but does not strictly abide by the rules of logic. Such relations useful for tutoring purposes were defined by comparing the definitions of concepts and inference rules in Omega and tracing common patterns among them. In particular, relations between mathematical concepts, relations between mathematical concepts and inference rules, as well as relations between concepts, formulae and inference rules were defined. We include examples of the mathematical knowledge for set theory that is represented in Omega in Appendix A. The additional relations for tutoring purposes that were defined for the Wizard-of-Oz experiment of the Dialog project and were later adjusted based on the experimental data are also in Appendix A. Moreover, Figure 3.2 gives an overview of the expanded ontology. It depicts the instructional points with their subclasses and instances. This enhanced ontology is an interface between `Menon` and the Domain Manager of Figure 1.1. The examples we use are for clarifying how this interface can be applied in this context and illustrate the mapping of the instructional points to the entities in a proof step.

## 3.4 Formalisation of Instructional Points

In this section, we define instructional points relevant to a proof and an inference step, based on the relations for tutoring purposes and as part of the enhanced ontology. The enhanced ontology has not been implemented as it depends on the overall mathematical knowledge representation which is not in the scope of `Menon`. `Menon` generates the abstract concepts (instructional points) which are to be instantiated by the concrete ontology in the Domain Information Manager that communicates with the Domain Reasoner for that purpose (see Figure 1.1). All names of inference rules and relations in Section 3.4 refer to the definitions in Appendix A.

### 3.4.1 Domain-Object Instructional Points

Let $E_s$ be the inference step in a particular proof, which comprises the premises $\varphi_1, \ldots, \varphi_n$, the conclusion $\psi$ and the Rule of Inference instructional point[3] P. Let $s$ be the source expression, and $t$ the target expression of $E_s$. Let also $\sigma$ and $\sigma'$ be mathematical concepts or expressions (e.g., logical quantifiers and connectives, the concepts of premise and conclusion, sets). We define the following instructional points:

[3] Defined in Section 3.4.3.2.

#### 3.4.1.1 Relevant Concept

**Relevant Concept:** $\sigma$ is a *Relevant Concept* for P in $E_s$ when:

1. *in*($\sigma$, $s$), and
2. *extracts*($\sigma$, P), and
3. *relevant-to*($\sigma$, P)

Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ fulfil the conditions of the definition of Relevant Concept. Then we choose one as the Relevant Concept, the one that appears less often in the premises. If more than one of $\sigma_1, \sigma_2, \ldots, \sigma_n$ appear equally often, but less often than others, then it suffices to choose one of them as the Relevant Concept arbitrarily[4].

Notation: *Rel-Con*($\sigma$, P, $s$), that is, $\sigma$ is the Relevant Concept in P in relation to $s$.

Examples: Let $E_s$ be Step 1 of the proof in Figure 1.4 and $s$ the *source* (i.e. $A \subseteq K(B) \Rightarrow B \subseteq K(A)$):
*Rel-Con*($\Rightarrow, \Rightarrow\text{-}I, s$), that is, the $\Rightarrow$ is the Relevant Concept for $\Rightarrow$-$I$ in relation to $s$.

Let $E_s$ be Step 2 of the proof in Figure 1.4, $s$ the *source* (i.e. $B \subseteq K(A)$): *Rel-Con*($\subseteq, Definition\ of\ Subset, s$), that is, the $\subseteq$ is the Relevant Concept for the *Definition of Subset* in relation to $s$.

More examples of Relevant Concepts for particular proof steps can be found in Section 6.6.

#### 3.4.1.2 Subordinate Concept

**Subordinate Concept:** $\sigma$ is a *Subordinate-Concept* for P in $E_s$ when:

1. *in*($\sigma$, $t$), and
2. *inserts*($\sigma$, P), and
   (a) either
      i. P is the definition of $\sigma$, and
      ii. *Rel-Con*($\sigma'$,P), and
      iii. *hypotaxon*($\sigma$, $\sigma'$),
   (b) or
      i. P is the definition of $\sigma'$, and

[4]If such $\sigma_1, \sigma_2, \ldots, \sigma_n$ existed, then this would mean that the inference rules that insert them can all be applied, giving rise to different proof steps. Since this instructional point is employed to help the student find the inference rule for the step, choosing one of them arbitrarily will just result in instructing the student to finding one of these inference rules.

ii. *Rel-Con*($\sigma'$,P), and

iii. *hypotaxon*($\sigma'$, $\sigma$)

(c) or *relevant-to*($\sigma$, P)

Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ fulfil the conditions of the definition of Subordinate Concept. Then we choose one as the Subordinate Concept, the one that appears less often in the conclusion. If more than one of $\sigma_1, \sigma_2, \ldots, \sigma_n$ appear equally often, but less often than others, then it suffices to choose one of them as the Subordinate Concept arbitrarily.

Notation: *Sub-Con*($\sigma, \mathrm{P}, t$), that is, $\sigma$ is the Subordinate Concept for P in $t$.

Examples: Let $E_s$ be Step 1 of the proof in Figure 1.4, $E_p$ be the part of the expression that has to be proved ($B \subseteq K(A)$ in $E_s$) and $t$ the *target* (i.e. assume $A \subseteq K(B)$ and prove $B \subseteq K(A)$):
*Sub-Con*($E_p, \Rightarrow\text{-}I, t$), that is the Subordinate Concept for $\Rightarrow$-$I$ is the expression that has to be proved in $t$.

Let $E_s$ be Step 2 of the proof in Figure 1.4 and $t$ the *target* (i.e. $\forall x : x \in B, x \in K(A)$):
*Sub-Con*($\in, Definition\ of\ Subset, t$), that is, $\in$ is the Subordinate Concept for the *Definition of Subset* in $t$.

More examples of Subordinate Concepts for particular proof steps can be found in Chapter 6, Section 6.6.

#### 3.4.1.3 Relevant-Concept Meta-Reasoning

Let $E_s$ be the expected inference step in a particular proof. Then this Relevant-Concept Meta-Reasoning is a relation between the Relevant Concept and $E_s$. That is, considering the Relevant Concept helps in finding the right inference rule for $E_s$.
Notation: *Rel-Con-Meta-Reas*($Rel\text{-}Con, E_s$)

Examples: Let $E_s$ be Step 1 of the proof in Figure 1.4:
*Rel-Con-Meta-Reas*($\Rightarrow, E_s$), that is, considering the $\Rightarrow$ helps in finding the right inference rule for $E_s$.

#### 3.4.1.4 Subordinate-Concept Meta-Reasoning

Let $E_s$ be the expected inference step in a particular proof. The Subordinate-Concept Meta-Reasoning is a relation between the Relevant Concept the Subordinate Concept and $E_s$. That is, considering the Relevant Concept in connection to the Subordinate Concept helps in finding the right inference rule for $E_s$.

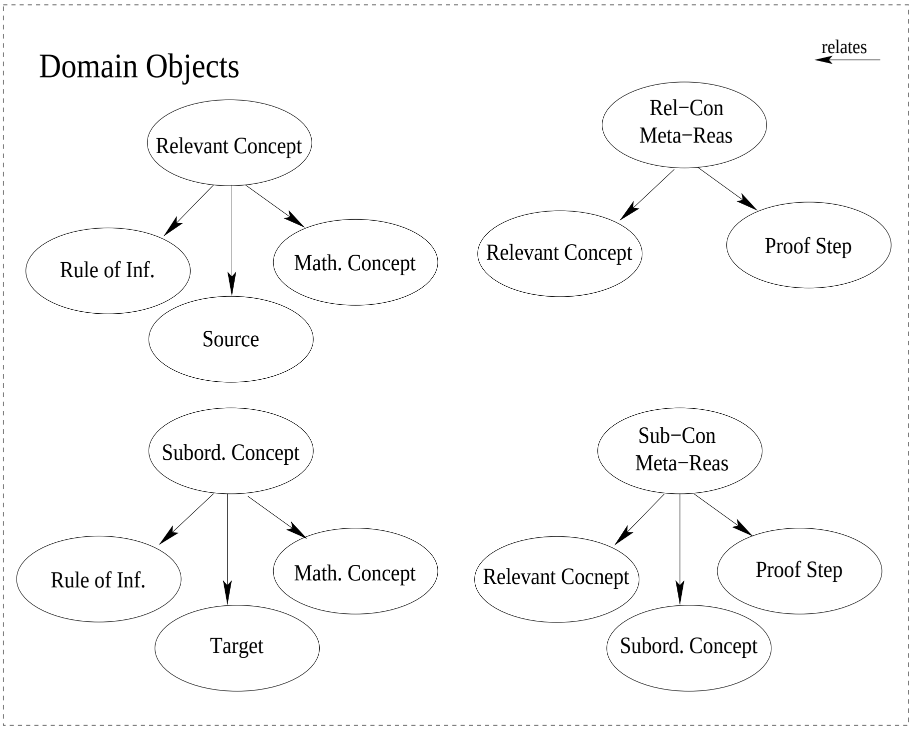


Figure 3.3: Domain-Objects Instructional Points

Notation: *Sub-Con-Meta-Reas*($Sub\text{-}Con, Rel\text{-}Con, E_s$)

<u>Examples:</u> Let $E_s$ be Step 1 of the proof in Figure 1.4 and $E_p$ the part of the expression that has to be proved in $E_s$:
*Sub-Con-Meta-Reas*($E_p, \Rightarrow, E_s$), that is, the expression $A \subseteq K(B) \Rightarrow B \subseteq K(A)$ includes $\Rightarrow$ (Relevant Concept). Therefore, one has to consider the part of the expression that needs to be proved (Subordinate Concept) in connection to $\Rightarrow$. The Subordinate Concept in this step is $B \subseteq K(A)$.

Let $E_s$ be Step 2 of the proof in Figure 1.4:
*Sub-Con-Meta-Reas*($\in, \subseteq, E_s$), that is because the mathematical expression $B \subseteq K(A)$ includes $\subseteq$ (Relevant Concept), one has to consider the $\in$ (Subordinate Concept) in connection to $\subseteq$.

Figure 3.3 gives an overview of the Domain-Object instructional points.

### 3.4.2 Domain-Relation Instructional Points

Let $E_s$ be the expected inference step in a particular proof, consisting of the premises $\varphi_1, \ldots, \varphi_n$, the conclusion $\psi$ and the rule P. A domain relation is a binary relation between a mathematical concept and a Domain Object, that is, the Relevant Concept or Subordinate Concept. Let $\sigma$ be a Domain Object and $\sigma'$ be another mathematical concept that is related to $\sigma$ with one of the interrelations of mathematical concepts, presented in Section A.2.4.1, namely: antithesis, duality, conversion, hypotaxis, primitive, specialisation, generalisation.
Notation: *Dom-Rel*$(\sigma, \sigma')$, that is, $\sigma$ is a Domain Relation to $\sigma'$ (cf. Figure 3.4).

Examples: In Step 2 of the proof in Figure 1.4, the Relevant Concept is $\subseteq$ and the Subordinate Concept is $\in$. If the student wanted to use $\not\subseteq$ instead of $\subseteq$, then this would be a antithetical concept. If the student wanted to use $\supseteq$ instead of $\subseteq$, then this would be a converse concept. If the student wanted to use $\in$ instead of $\subseteq$, that would be a hypotaxon. If the student wanted to use $\subset$ instead of $\subseteq$, that would be a specialisation. By informing students of such relations we try to make them more aware of the structure of the domain, and help them draw the necessary inference for the step.

We look at examples of how Domain Relations can be used in tutoring in Section 6.6.

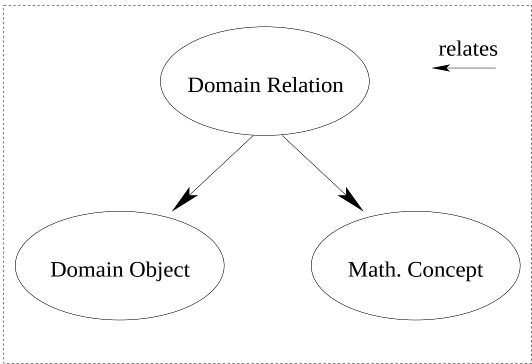


Figure 3.4: Domain-Relation Instructional Points

### 3.4.3 Rule-of-Inference Instructional Points

The instructional point Rule of Inference includes the justification in terms of inference rule and domain technique for a proof step. The general pedagogical notion of a *Rule of Inference* is the rule that justifies the current reasoning step. That is, it is a schema for deriving a formula, the *conclusion*, from other formulae, the *premises*. It can be applied *forwards* to infer the conclusion from the given premises, or *backwards* to obtain the premises needed to infer a desired conclusion.

#### 3.4.3.1 Domain Technique

Let $E_s$ be the expected inference step in a particular proof, P the instructional point Rule of Inferences, $R_c$ the Relevant Concept in $E_s$, and $D$ is either the domain technique that the Rule of Inference uses (i.e. case distinction or induction) or else the occurrence state of the Relevant Concept that the application of P influences. Then Domain Technique is a relation between $D$, P and $R_c$, that is, that $D$ is the Domain Technique that P uses to handle $R_c$.
Notation: *Dom-Tech*$(D, R_c, \mathrm{P})$.

Examples: Let $E_s$ be Step 1 of the proof in Figure 1.4:
*Dom-Tech*$(extract, \Rightarrow, \Rightarrow\text{-}I)$, that is the Domain Technique that $\Rightarrow\text{-}I$ uses for $\Rightarrow$ is *extract*.

We use *occurrence state* as the Domain Technique in place of elimination and introduction of defined constants, quantifiers and connectives for pedagogical purposes. Namely, to allow students to extract useful patterns and apply forward reasoning as much as possible. Any reference to the logical concepts of elimination and introduction has to make use of backward and forward application of inference rules that is counter-intuitive and adds extra cognitive load (cf. Chapter 2). We leave the acquisition of any other necessary knowledge to implicit learning. The choice between reducing the cognitive load and instructing the student based on logical concepts is not an obvious one and it might be unsuitable. However, in such a complicated domain as proving (see, for example, the complexity of Figure 3.2) implicit learning is an important tool for learning (cf. Chapter 2).

The Domain Techniques that we consider in our domain are the following:

**Occurrence state of defined constants** Elimination and introduction of predicates and functions, e.g. definitions, theorems, lemmata, tautologies.

Examples: For the forward application of the rule $\mathcal{P}\text{-}I$:
*Dom-Tech*$(insert, \mathcal{P}, \mathcal{P}\text{-}I)$, that is, the Domain Technique in $\mathcal{P}\text{-}I$ relevant to the $\mathcal{P}$ is *insert*.

**Case distinction** Case distinction is applied when we have a disjunctive definition. The cases are then the terms of the disjunction.

Examples: For case distinction applied for the *Definition of Union* $(A \cup B = \{x | x \in A \ or \ x \in B\})$:
*DomTech*(*case distinction*, $\cup$, *Definition of Union*), that is, the Domain Technique in the *Definition of Union* relevant to the $\cup$ is case distinction.

**Induction** Induction is applied, when we have an inductive definition. The inductive cases are the base step, the inductive hypothesis and the inductive step.

Examples: Let's assume the following inductive definition: $P_f(X)$ is the set of all finite subsets of X if: (i) $\emptyset \in P_f(X)$ (ii) $A \in P_f(X), x \in X \Rightarrow A \cup \{x\} \in P_f(X)$.

For induction applied to the definition of $P_f$:
*Dom-Tech*(*induction*, $P_f$, *definition of the* $P_f$), that is, the Domain Technique in the *definition of* $P_f$ relevant to the $P_f$ is induction.

Case distinction and induction are considered as Domain Techniques. The cases that realise the case distinction or the induction are the instances of the instructional point Rule of Inference each time.

**Occurrence state of quantifiers** Elimination and introduction of universal and existential quantifiers.

Examples: For the forward application of the rule $\forall$-$E$:
*Dom-Tech*(*extract*, $\forall$, $\forall$-$E$), that is, the Domain Technique in $\forall$-$E$ relevant to $\forall$ is *extract*.

**Occurrence state of connectives** Elimination and introduction of conjunction, disjunction, implication, equivalence, and equation.

Examples: For the forward application of the rule $\wedge$-$I$:
*Dom-Tech*(*insert*, $\wedge$,$\wedge$-$I$), that is the Domain Technique in $\wedge$-$I$ relevant to $\wedge$ is *insert*.

More examples of Domain Techniques for particular proof steps can be found in Section 6.6.

#### 3.4.3.2 Rules of Inference

Let $E_s$ be the expected inference step in a particular proof, consisting of the premises $\varphi_1, \ldots, \varphi_n$, the conclusion $\psi$ and the rule P. The instructional point Rule of Inference is the rule P used as the justification for making the inference in $E_s$.
Notation: *Rule-of-Inf*$(\mathrm{P}, E_s)$.

Examples: Let $E_s$ be Step 1 of the proof in Figure 1.4:
*Rule-of-Inf*$(\Rightarrow\text{-}I, E_s)$, that is $\Rightarrow$-$I$ is the instructional point Rule of Inference in $E_s$.

We subdivide the instructional point Rule of Inference based on the Domain Technique that Rules of Inference involve. This categorisation aims at capturing the underlying deep understanding of experts and is based on the structure suggested by Schreiner [Schreiner, 2004] for formal proving.

Let us see how the Domain Techniques are related to the instructional point Rules of Inference.

**Occurrence state of defined constants, substitution, etc:** The definitions in Appendix A are the Rule of Inference instructional points considered under this Domain Technique.

Examples: *Commutativity of Union*
*Definition of Subset*

**Case distinction:** The cases of the case distinction are the Rule of Inference instructional points.

Examples: For the case distinction of the *Definition of Union* ($A \cup B = \{x | x \in A$ *or* $x \in B\}$), the Rule of Inference instructional points are the cases: $x \in A$ or $x \in B$.

**Induction:** The inductive steps are the Rule of Inference instructional points.

Examples: For induction applied to the definition of $P_f$ ($P_f(X)$ *is the set of all finite subsets of* $X$ *if: (i)* $\emptyset \in P_f(X)$ *(ii)* $A \in P_f(X), x \in X \Rightarrow A \cup \{x\} \in P_f(X)$) if we want to prove that $\Phi(B)$ holds for all $B \in P_f(X)$, the steps are (i) $\Phi(\emptyset)$ and (ii) for $A \in P_f(X)$ and $x \in X$ if $\Phi(A)$ holds, then $\Phi(A \cup \{x\})$ holds.

Note that in case distinction as well as in induction, once the step is completed, the next step becomes simply the first case or step to be proved. Hinting is based on that.

**Occurrence state of quantifiers:** This includes the rules for the universal and the existential quantifiers, presented in Appendix A.

Examples: $\forall$-$E$
$\forall$-$I$

**Occurrence state of connectives:** This includes the rules for conjunction, disjunction, implication, equivalence, and equality that are included in Section A. These logical connectives share the same pattern for instantiating the Domain Objects relevant to them (cf. instructional point in Section 3.4.1).

Examples: $\wedge$-$I$
$\wedge$-$E_L$
$\wedge$-$E_R$

More examples of Rule of Inference instructional points for particular proof steps can be found in Section 6.6.

#### 3.4.3.3 Connect Relevant-Subordinate-Concept

Let $E_s$ be the expected inference step in a particular proof, P the instructional point Rule of Inference, $R_c$ the Relevant Concept, and $S_c$ the Subordinate Concept (cf. Section 3.4.1). Then, Connect Relevant-Subordinate-Concept is a relation between $R_c$, $S_c$ and P. Namely, that the application of P in $E_s$ involves $R_c$ and $S_c$.
Notation: *Con-Rel-Sub*$(R_c, S_c, \mathrm{P})$.

Examples: Let $E_s$ be Step 1 of the proof in Figure 1.4 and $E_p$ the part of the expression that needs to be proved, which is determined by the application of $\Rightarrow$-$I$:
*Con-Rel-Sub*$(\Rightarrow, E_p, \Rightarrow\text{-}I)$, that is the $\Rightarrow$-$I$ connects the $\Rightarrow$ and the part of the expression that needs to be proved, which is $B \subseteq K(A)$ in Step 1.

Let $E_s$ be Step 2 of the proof in Figure 1.4: *Con-Rel-Sub*($\subseteq$, $\in$, *Definition of Subset*), that is, the *Definition of Subset* connects the $\subseteq$ and the $\in$.

#### 3.4.3.4 Elaborate Domain Object

Let $E_s$ be the expected inference step in a particular proof, P the instructional point Rule of Inference, $R_c$ the Relevant Concept, and $D$ the Domain Technique. Then, this is a relation between P, $R_c$ and $D$ in $E_s$. Namely, that P handles $R_c$ via $D$. Notation: *Elab-Dom-Obj*$(\mathrm{P}, D, R_c)$.

Examples: For Step 1 of the proof in Figure 1.4:
*Elab-Dom-Obj*$(\Rightarrow\text{-}I, extracts, \Rightarrow)$, that is the $\Rightarrow$-$I$ extracts the $\Rightarrow$.

Figure 3.5 gives an overview of the Rule-of-Inference instructional points.

### 3.4.4 Substitution Instructional Points

#### 3.4.4.1 Substitution

Let $E_s$ be the expected inference step in a particular proof, consisting of the premises $\varphi_1, \ldots, \varphi_n$, the conclusion $\psi$ and the instructional point Rule of Inference P. The Substitution is a relation between $\varphi_1, \ldots, \varphi_n$, $\psi$ and P, that is the substitution for P is done by replacing premises and the conclusion of P with the relevant parts of the expression at hand in $E_s$.
Notation: *Substitution*$(\mathrm{P}, \varphi_1, \ldots, \varphi_n, \psi)$.

Examples: In Step 1 of the proof in Figure 1.4:
*Substitution*$(\Rightarrow\text{-}I, A \subseteq K(B), B \subseteq K(A))$, that is, the substitution for $\Rightarrow$-$I$ is done by replacing $A$ with $A \subseteq K(B)$ and $B$ with $B \subseteq K(A)$.

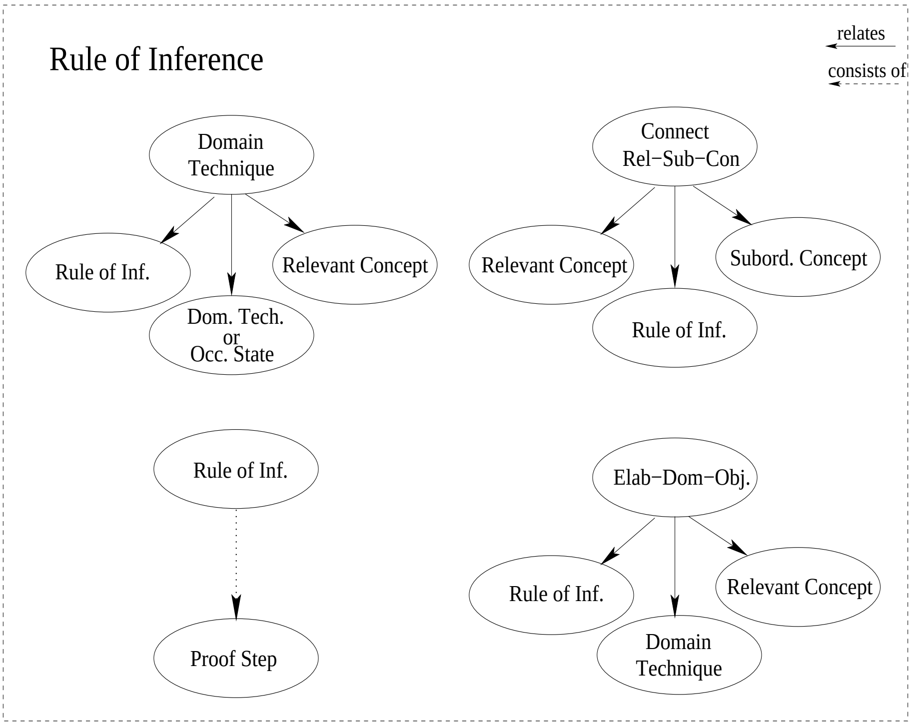


Figure 3.5: Rule-of-Inference Instructional Points

#### 3.4.4.2 Rule-of-Inference Application

Let $E_s$ be the expected inference step in a particular proof, $D$ the Domain Technique, $R_c$ the Relevant Concept, and P the instructional point Rule of Inference in $E_s$. Then Rule-of-Inference Application is a relation between $D$, $R_c$, and P. Namely, using P to apply $D$ on $R_c$ is the substitution in $E_s$.
Notation: *Rule-of-Inf-Appl*$(D, R_c, \mathrm{P},$ *Substitution*$)$.

Examples: Let $E_s$ be Step 1 of the proof in Figure 1.4 and $S$ be the substitution in $E_s$:
*Rule-of-Inf-Appl*$(extract, \Rightarrow, \Rightarrow\text{-}I, S)$, that is the $\Rightarrow$ has to be extracted by $\Rightarrow$-$I$ so that we assume $A \subseteq K(B)$ and prove $B \subseteq K(A)$ (Substitution).

Figure 3.6 gives an overview of the Substitution instructional points.

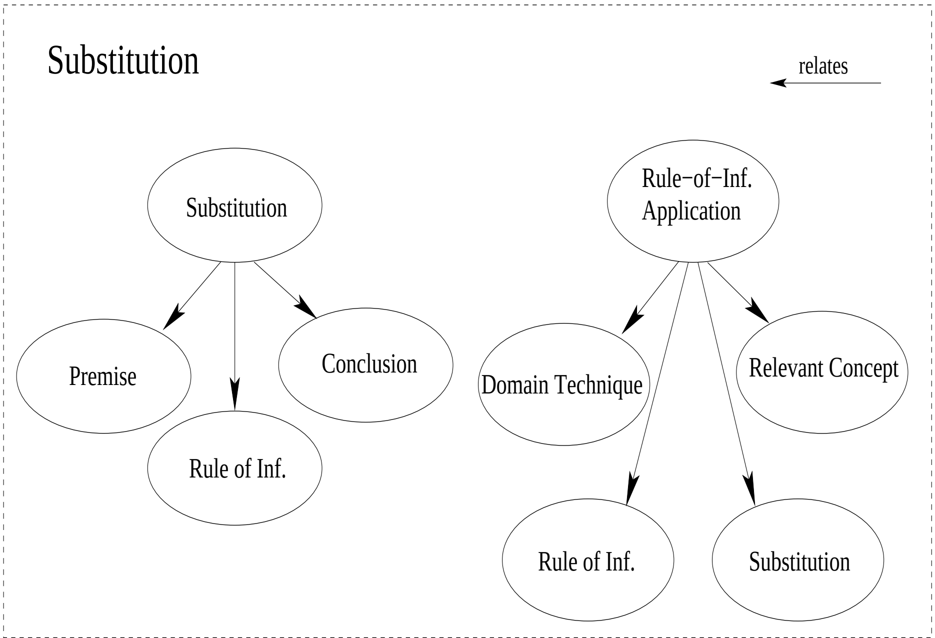


Figure 3.6: Substitution Instructional Points

### 3.4.5 Proof-Step Instructional Points

The Proof Step instructional point consists of the premises, the conclusion and the applied Rule of Inference instructional point. It also comprises proof methods such as proof by contradiction (indirect proof). Let $E_s$ be the expected inference step in a particular proof, consisting of the premises $\varphi_1, \ldots, \varphi_n$, the conclusion $\psi$ and the instructional point Rule of Inference P. We define the following:

#### 3.4.5.1 Proof Step

Let $E_x$ be the mathematical expression that we want to prove. Proof Step includes three instructional points.

**Premise-Conclusion:** These are the premises $\varphi_1, \ldots, \varphi_n$, and the conclusion $\psi$ in $E_s$.
Notation: *Prem-Conc*$(\varphi_1, \ldots, \varphi_n, \psi, E_s)$.

Examples: In Step 1 of the proof in Figure 1.4, the premise is $B \subseteq K(A)$ under hypothesis $A \subseteq K(B)$ and the conclusion $A \subseteq K(B) \Rightarrow B \subseteq K(A)$.

This instructional point deals with the premises and the conclusion together, as one cannot reason about the one without at the same time reasoning about the other. Also, dealing with them together allows us to use *source* and *target* for instruction without having to differentiate between premise and conclusion that is often an unintuitive distinction, for example in backward steps.

**Starting Point:** The Starting Point is a relation between $\varphi_1, \ldots, \varphi_n$, $\psi$, and expression that we want to prove ($E_x$). It represents that the reasoning about $E_s$ should start by identifying the $\varphi_1, \ldots, \varphi_n$ and $\psi$.
Notation: *Start-Point*(*Prem-Conc*, $E_x$).

Examples: In the proof in Figure 1.4:
*Start-Point*(*Prem-Conc*, $A \subseteq K(B) \Rightarrow B \subseteq K(A)$), that is, the reasoning has to start from identifying the premise and the conclusion of $A \subseteq K(B) \Rightarrow B \subseteq K(A)$.

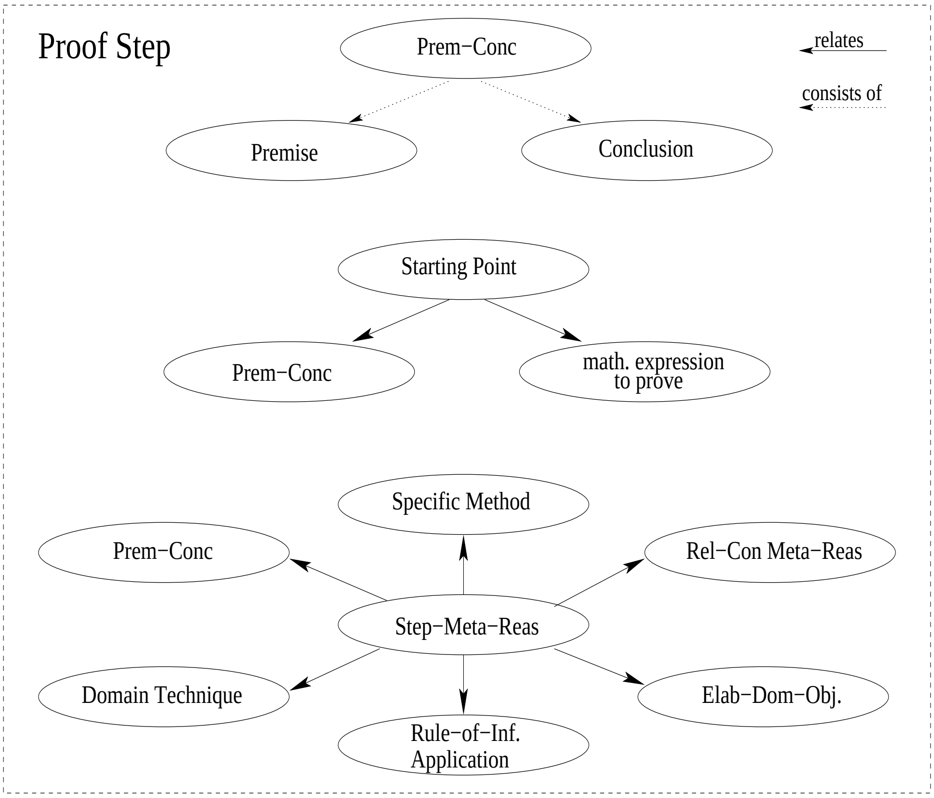


Figure 3.7: Proof Step

**Step-Meta-Reasoning:** The Step-Meta-Reasoning is a relation representing the connection among the different instructional points in $E_s$, namely that all together they amount to the derivation of $E_s$.
Notation: *Step-Meta-Reas*(*Prem-Conc*,*Spec-Method*,*Rel-Con-Meta-Reas*, *Dom-Tech*,*Elab-Dom-Obj*,*Rule-of-Inf-Appl*, $E_s$)

Examples:Let $E_s$ be Step 1 of the proof in Figure 1.4, P-C the Premise-Conclusion, $S_p$ the Specific Method, $RC_m$ the Relevant-Concept Meta-Reasoning, $D$ the Domain Technique, $E_{do}$ the Elaborate Domain Object, and $\mathrm{P}_a$ the instructional point Rule-of-Inference Application, all in $E_s$:
*Step-Meta-Reas*(P-C,$S_p, RC_m, D, E_{do}, \mathrm{P}_a, E_s$), that is, first we identify the premise $A \subseteq K(B)$ and the conclusion $B \subseteq K(A)$. Then we consider the $\Rightarrow$ because it helps us simplify the step, and we apply the rule $\Rightarrow$-$I$ to extract the $\Rightarrow$, by assuming $A \subseteq K(B)$ and proving $B \subseteq K(A)$.

We will see more examples of this instructional point and how it is used in tutoring in Section 6.6.

Figure 3.7 gives an overview of the Proof-Step instructional points.

#### 3.4.5.2 Proof Method

Let $P$ be a proof consisting of its proof steps $P_1, \ldots, P_n$. Proof Method includes two instructional points.

**Abstract Method:** The Abstract Method is the most abstract method applied in the proof, which in our domain is the proof directness $P_d$.
Notation: *Abs-Method*($P_d, P$).

We consider:

- Direct Proof: Assuming the premises and proving conclusion.
  Examples: Let $P$ be the proof in Figure 1.4:
  *Abs-Method*(*direct-proof*, P), that is, the Abstract Method in $P$ is *direct proof* (we assume the premise $A \subseteq K(B)$ and prove from that the conclusion $B \subseteq K(A)$).
- Indirect Proof: Assuming the negation of the conclusion and proving a contradiction.
  Examples: Let $P$ be the following proof, which is a proof by contradiction for the same task as in Figure 1.4 (namely, if $A \subseteq K(B) \Rightarrow B \subseteq K(A)$)
  1. Let $A \subseteq K(B)$, we will show that $B \not\subseteq K(A)$.
  2. Let $x$ such that $x \in B$ and $x \notin K(A)$.
  3. Then $x \in A$.
  4. If $x \in A$, then $x \in K(B)$.

5. But then $x \in K(B)$ and $x \in B$, which is a contradiction.
6. Therefore if $x \in B$, then $x \in K(A)$.
7. From that it follows that $B \subseteq K(A)$.

Then, *Abs-Method*(*indirect-proof*, P), that is, the Abstract Method in $P$ is *indirect proof* (we assume the opposite of $B \subseteq K(A)$ and we prove a contradiction).

**Specific Method:** The Specific Method is the specific method applied in the proof, which in our domain is the proof direction $P_r$.
Notation: *Spec-Method*$(P_r, E_s)$ (cf. Figure 3.8)

We consider:

- Forward step: A Proof Step where new knowledge is derived from known facts. The method is applying rules to manipulate the expression towards the goal.

  Examples: Let $E_s$ be Step 4 of the proof in Figure 1.4:
  *Spec-Method*$(forward, E_s)$, that is, the Specific Method in $E_s$ is *forward* (we apply the inference rule *Definition of Complement* to manipulate the expression towards the goal $A \subseteq K(B) \Rightarrow B \subseteq K(A)$).
- Backward step: A step where the goal is decomposed and the reasoning starts from a hypothesis. The method is decomposing the goal.

  Examples: Let $E_s$ be Step 3 of the proof in Figure 1.4:
  *Spec-Method*$(backward, E_s)$, that is, the Specific Method in $E_s$ is *backward* (we decompose the goal $B \subseteq K(A)$ by assuming that $x \in B$ and showing that $x \in K(A)$).

Figure 3.5 gives an overview of the Proof-Method instructional points.

### 3.4.6 Use of the Domain Ontology and the Instructional Points

The overall purpose of the ontology is to enable the automation of the Socratic teaching-model. As shown in Figure 3.9, the ontology was developed for use in the analysis of the student answer. This analysis should identify which instructional points are used in the student answer, and thus capture the students' level of performance and their ability to apply the domain knowledge for the task. In this chapter we saw how the domain ontology defines the instructional points that are useful for tutoring in general and for learning how to prove in particular. We represent this analysis of the student answer in the (HSS), which constitutes the input to the teaching strategy (cf. Chapter 5).

The representation of the domain ontology in the HSS is additionally evoked by the Socratic teaching strategy in the determination of the hint category to

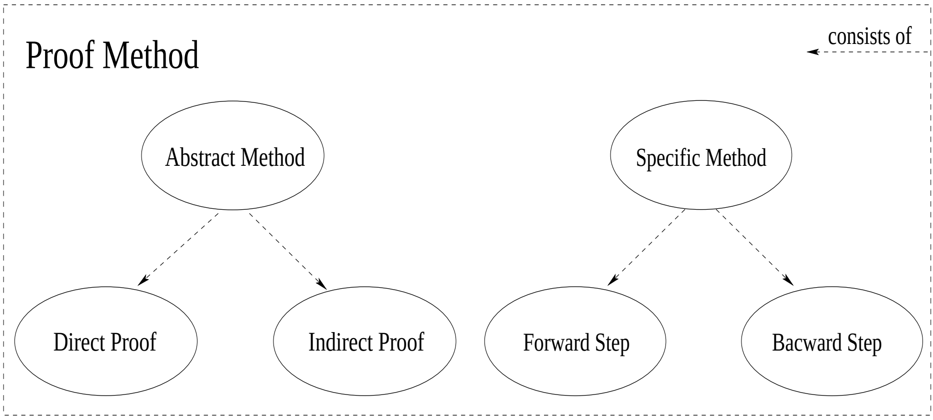


Figure 3.8: Proof Method

be produced. Our goal is to dynamically produce hints that fit the cognitive needs of the student with regard to the particular proof. However, we want to be able to do that for arbitrary proofs in a domain. Therefore we cannot restrict ourselves to a gamut of static hints which is compiled by associating elements of the student answer with a unique response by the system. As an alternative to such approaches, we use the definition of hint categories based on instructional points. The role of the ontology is to assist in mapping the instructional points onto the actual objects or relations that are used in the particular context. That means that once the hint category has been chosen, the domain ontology captures the necessary instantiations of instructional points for the particular proof and the proof step under consideration. This is the actual information that `Menon` provides for the natural language realisation of the hint category (cf. Chapter 6). Our ontology is a first step towards an infrastructure for plugging in any domain reasoner that can provide the needed domain-specific information and converting it into a database that includes the relations that are necessary for hinting.

Moreover, the motivation for this work derives from the need to formalise the cognitive functions that underly hints, in order to produce adequate and psychologically justified feedback. One of our central goals is also to separate out such underlying functions of hints from dialogue-move functions, which might be common for different cognitive functions. The domain ontology is a tool to this end, singling out and representing cognitive aspects of hints via instructional points to promote the acquisition of schemata. Namely, the abstract definitions of instructional points derived in Section 3.2 are used to define the domain-information content of hints that is a cognitive function. The ontology, in turn,

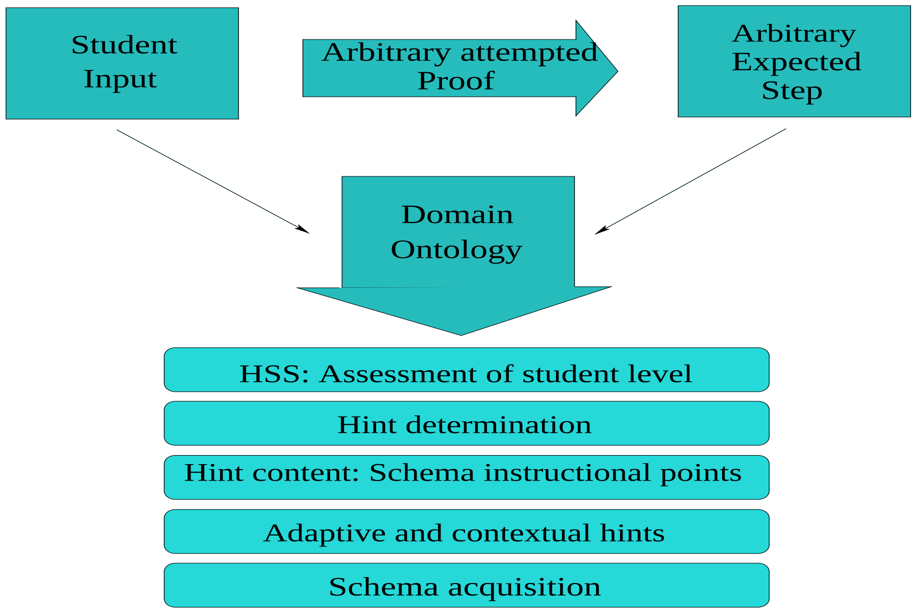


Figure 3.9: The Use of the Ontology in Automating NL Hinting for Schema Acquisition

enables the instantiation of these abstract instructional points specifically for the proof and proof step at hand.

The instructional points facilitate the acquisition of schemata for problem solving in the domain (cf. Chapter 2). The definition of the instructional points is also the first step towards the definition of hint specifications. The specifications are complemented by additional tutoring information, which is captured in other dimensions of the hint taxonomy. This completes the task-related information. The task information can then be further enhanced by dialogue and discourse information for the dialogue in progress. All of this information together provides the possibility of adaptive and contextualised hint realisation (cf. Chapter 4).

# Chapter 4

# Dialogue Moves and Hints

## 4.1 Introduction

In this chapter we present an analysis of dialogue moves for tutorial dialogues captured in a taxonomy of six dimensions [Tsovaltzi and Karagjosova, 2004]. We concentrate on the dimensions TASK, CONVENTIONAL TASK-MANAGEMENT, and CONVENTIONAL COMMUNICATION-MANAGEMENT, as the moves in these dimensions are the ones manipulated in `Menon`. The rest of the dimensions and the complete background of the dialogue-move taxonomy can be found in Appendix C. As the focus of our analysis, we provide an investigation of the move *hint*. This investigation results itself in a multidimensional hint taxonomy, which supports the automatic production and natural language realisation of user-adaptive hints for arbitrary tasks.

## 4.2 Dialogue Move Taxonomy

We extend standard dialogue-move taxonomies for the genre of tutorial dialogues [Tsovaltzi, 2001] and develop a dialogue-move taxonomy based on an analysis of our empirical data on tutoring set theory [Wolska *et al.*, 2004]. The taxonomy draws on DAMSL [Allen and Core, 1997], which is an attempt to provide a standard top-level structure for annotating dialogues for reusable annotation schemes. We make use of the DAMSL multiple-level structure, which allows us to cater for the various functions an utterance may have in dialogue.

We modified and extended the dialogue moves in DAMSL with moves from the BE&E (Basic Electricity & Electronics) annotation scheme [Core *et al.*, 2002], which was developed for the tutorial-dialogue genre. The BE&E annotation scheme is based itself on DAMSL, however without distinguishing all levels that DAMSL provides for. Furthermore, it provides additional dialogue moves, derived from a tutorial-dialogue corpus. We adopt some of these moves, but in order to cater for more genre-specific (tutorial dialogue) and domain specific (e.g. proving in set theory) phenomena, we define new dialogue moves and group them in a separate task level, following DAMSL. The need for such separation in dialogue systems has also been advocated by [Allen *et al.*, 2001b] who use a modular architecture for dialogue and task planning for the domain of route planning. Moreover, [Zinn, 2002] argues for its advantages in the tutorial-dialogue genre.

### 4.2.1 General Description of the Dialogue Move Taxonomy

Our taxonomy features six dimensions, one for every function that an utterance might have. The actual detailed description of what the utterance effects in the dialogue corresponds to one of the categories of dialogue moves. In every dimension, there are different levels of description of the dialogue moves. These are structured in classes and subclasses, up to three levels deep. The six dimensions in the taxonomy are:

1. Forward Looking
2. Backward Looking
3. Task
4. Conventional task-management
5. Conventional communication-management
6. Communicative status

We propose an expanded task dimension and define dialogue moves in a bottom-up manner using our empirical data (cf. [Wolska *et al.*, 2004]). We are additionally guided by tutorial-dialogue specific moves defined in [Core *et al.*, 2002], as well as by psychological considerations regarding teaching [Wilson and Cole, 1991; Lim and Moore, 2002; Weiner, 1992]. We claim that adding the task dimension helps to separate generic dialogue management, from manipulating genre and domain specific phenomena, involved in modelling different teaching models and domains. This provides a better framework for capturing what is generic in dialogue management and, hence, reusable between genres, from what is specific to the genre or the domains. modelling the genre and the

In such a framework, tutorial systems can leverage the advantages of NL capabilities in tutoring, which have been discussed in [Moore, 1993], independently from their preferred teaching strategies. Teaching models that presuppose dialogue interaction have been demonstrated to be a beneficial ingredient of tutoring [Chi *et al.*, 1994; Rosé *et al.*, 2001b]. Additionally, the modelling of such dialogue-based teaching strategies can help manipulate psychological aspects of learning, such as help the student build a deeper understanding of the domain, eliminate cognitive load, and promote schema acquisition [Wilson and Cole, 1991; Lim and Moore, 2002].

In our expanded task dimension, task-level moves are clearly defined for use in tutorial dialogues. Such definitions facilitate our understanding of the phenomena in tutorial dialogues. Task-dimension dialogue moves are often indirect ones and acquire their specific function only due to the tutorial genre or require special treatment in the genre. They can also only be planned by use of special modules that deal with domain knowledge (cf. [Benzmüller *et al.*, 2003a; Buckley and Benzmüller, 2005]) or after consulting information on the current student progress. For instance, before any reasonable feedback can be given to the student, the student's input has to be evaluated. There are, however, two categories of student input that are relevant to tutoring. There is input that does not contribute to the task but carries information relevant to tutoring, such as the level of motivation of the student. This aspect of the input is useful for the genre irrespective of the teaching model implemented. For example, it is useful to represent the move *resign* (Example 4.1)[1], i.e., the student stating that she does not wish to continue with the task. The way this move will be treated can be decided according to the pedagogical model of choice. To name two opposing cases, it may cause the tutor to give away the solution to the task or to use some strategy that would motivate the student to move on again.

(4.1) **S3:** "Gebe auf."
*[I give up.]*

On the other hand, there is task-related input with which the student tries to handle the task. It is characteristic of this kind of input, first, that it is domain-specific and, second, that it needs to be evaluated with respect to its success in bringing the task forward. More specifically, the dialogue move *domain-contribution* must be available in order to represent attempts to bring the task forward. Once this is represented, it is up to the implementor to decide on categories for the evaluation of this attempt, that is the *domain-contributions* relevant to the tutoring purposes. This decision is both domain dependent (different domains might require divergent evaluation methods) and teaching-strategy dependent (the information useful for tutoring may vary between models). Hence, it lies beyond the scope of dialogue management. Chapters 5 and 6 deal with such formalisations, which are external resources for the dialogue manager.

In general, a clear distinction with regard to dialogue-move functions, like the one we are suggesting, offers itself as a valuable basis for specifying interfaces between the dialogue manager and other modules in a dialogue system (cf. Chapters 1 and 6).

[1]The examples are from the DIALOG corpus if not otherwise indicated. The reference name of the particular dialogue signifies, from right to left, the teaching method, the number of the subject and the proof that constituted the task. We provide the turn number, where relevant, next to the indication of who is uttering the turn, e.g., **T**: tutor, and **S**: student.

## 4.3 Dialogue-Move Analysis

In this section, we show the dimensions that are relevant to tutorial dialogues based on the theory of *obligations* [Traum and Allen, 1994; Matheson *et al.*, 2000; Tsovaltzi and Matheson, 2002; Tsovaltzi, 2001]. This theory introduces the notion of discourse and social obligation as a way of analysing some of the social aspects of interactions and provides an explanation for behaviour that other theories do not predict. It is an augmentation of the representation of the *intentions* [Rich and Sidner, 1998] of dialogue participants that attempts to capture the natural flow of conversation.

Treating tutorial dialogues in terms of obligations is a way of analysing and predicting some specific kinds of dialogue behaviour that do not seem to follow the rules of everyday discourse. For example, SharedPlans [Grosz and Sidner, 1986], which is a predominant model in the intentions tradition, do not explain why the tutor does not just give all answers away to the student. This violates the principle of co-operativity, which is central in such intention-based theories.

According to the obligation-driven framework, on the other hand, intentions are necessary but not the only driving force behind an utterance. For example, only if one considers it the students' obligation to address the tutor's questions and follow her directives is it possible to interpret the total lack of overt signals from the students that they intend to cooperate. Such signals are normally central to collaboration. In the context of the overall obligations the tutor knows what the students' intentions are, and will be able to interpret their actions correctly, because the tutorial-dialogue genre does not permit any other behaviour. That means that it is the students' obligation to follow the tutor's directives.

In the following, for each of the dialogue moves explored we include and intuitive definition, specifications, obligations, NL examples, relations to moves in other dimensions, and extra notes wherever these apply.

### 4.3.1 Conventional Task-Management Dialogue Moves

This dimension captures utterances that explicitly address the management of the structure of the task. It also includes implicit dialogue moves. In DAMSL tutoring-task and conventional task-management are collapsed into one dimension called "task management". As we have explained though, the tutoring task is way too important in the tutorial-dialogue genre and it deserves a separate dimension that makes the manipulations necessary in it clear. Moreover, although those manipulations might be contentious and subject to many changes to cater for different teaching models, conventional task management is much more straightforward and need not be affected by changes at the tutoring-task level. This makes reconfigurability easier. Therefore, we break the task management into two dimensions: TUTORING-TASK and CONVENTIONAL TASK-MANAGEMENT.

In [Allen and Core, 1997], this dimension captures utterances that "explicitly address the problem solving process and experimental procedure", including utterances that involve coordinating the activities of the interlocutors ("Let's work on getting the train to Avon first"), asking for help on the procedures ("Do I need to state the problem?") or asking about the status of the process ("Are we done?"). DAMSL distinguishes between utterances that are part of the task, and utterances involving the problem solving process (task management)[2]. We capture the following dialogue moves *initiate-task*, *close-task*, *initiate-subtask* and *close-subtask* in order to add the necessary discourse structure which corresponds to the tutoring-task structure and make it easier for the student to follow the purpose of subtasks. In consequence, *initiate-subtask* and *close-subtask* have to be dynamically realised for the current subtask to take the discourse structure into account and maximise coherence.

There are no obligations in this dimension. Only NL specifications are applicable based on the dialogue-move category as, for example, noone expects a user to reply to a greeting from the computer.

#### 4.3.1.1 Initiate-task

An utterance with which the task is initiated.

**NL Examples**

(4.2) **T1:** "Bitte zeigen Sie:..."
*[Please, show:...]* (soc5k)

(4.3) **T:** "OK, let's look at a proof! Tell me everything you can think of for proving the following:..." (constructed example)

**Relation to moves in other dimensions** *Initiate-task* can be at the same time also *initiate-dialogue* if there is no explicit *initiate-dialogue*.

**Notes** In principle, all *domain-contributions* address this utterance in addition to whichever utterance their backward-looking function addresses.

#### 4.3.1.2 Close-task

An utterance that indicates that the task is completed/closed.

**NL Examples**

(4.4) **T4:** "Damit ist die Aussage schon bewiesen."
*[With that, the expression is already proven.]* (did15d)

**Relation to moves in other dimensions** *Close-task* can be at the same time also *close-dialogue* if there is no explicit *close-dialogue*.

[2] Note that we have two tasks, proof task and tutoring task (cf. Section 4.4.2).

**Notes** The last move of the hinting session may also serve as *close-task* (e.g., did15p T8).

#### 4.3.1.3 Initiate-subtask

An utterance with which a subtask is initiated.

**NL Examples(constructed)**

(4.5) **T:** "Let's look at it"

#### 4.3.1.4 Close-subtask (constructed)

An utterance that indicates that a subtask is completed/closed.

**NL Examples**

(4.6) **T:** "OK"

(4.7) **T:** "OK, so much for that"

### 4.3.2 Conventional Communication-Management Dialogue Moves

This dimension concerns utterances that explicitly manage the structure of the dialogue. It comprises the following moves: *Initiate-dialogue*, *close-dialogue*, *initiate-subdialogue*,*close-subdialogue*, *discourse-marking*. These moves have also a forward-looking function, that is *conventional-opening* or *conventional-closing*.

There are no obligations in this dimension, which is a fact special to human-computer interaction. Only NL specifications are applicable based on the dialogue-move category.

#### 4.3.2.1 Initiate-dialogue

An utterance with which the dialogue is initiated.

**NL Examples (constructed)**

(4.8) **T:** "Hello!"

**Relation to moves in other dimensions** It can be implicit in the utterance with which the task is initiated, namely *initiate-task*.

#### 4.3.2.2 Close-dialogue

An utterance that indicates that the dialogue is finished/closed.

**NL Examples (constructed)**

(4.9) **T:** "Bye!"

**Relation to moves in other dimensions** It can be implicit in the utterance that closes the task, namely *close-dialogue*.

#### 4.3.2.3 Initiate-subdialogue

An utterance with which a subdialogue is initiated.

It is commonly a *request-clarification*, or a *signal-non-understanding* in the BACKWARD-LOOKING dimension.

**NL Examples (constructed)**

(4.10) **T:** "Let's see..."

**Notes** *Initiate-subdialogue* and *initiate-subtask* may coincide in the same way that *initiate-dialogue* and *initiate-task* may.

#### 4.3.2.4 Close-subdialogue

An utterance with which a subdialogue is ended/closed.

**NL Examples (constructed)**

(4.11) **T:** "So..."

#### 4.3.2.5 Discourse-marking

Utterances that mark the discourse structure, i.e., indicate how the current utterance relates to the preceding or succeeding context.

**NL Examples**

(4.12) **T4:** "Dies mache ich nun."
*[I'm going to do that then]* (did16k)

**Relation to moves in other dimensions** *Discourse-marking* refers to utterances as opposed to subdialogues or subtasks as *initiate-subtask* and *initiate-subdialogue*.

**Notes** *Discourse-marking* is not handled in `Menon`, as it relates to discourse structure.

## 4.4 Task Dimension

The task dimension is divided into two subdimensions in order to distinguish between the two tasks that are performed while tutoring, namely the proof task and tutoring task, and make their dialogue management transparent. The *proof task* is concerned with resolving the domain task for the session. In our domain the task is finding a mathematical proof to a problem. The *tutoring task* subdimension includes dialogue moves that do not try to resolve the task directly. This captures the intuition that it is not the task of the tutor to find a solution, but rather to help the student find it. This subdimension constitutes the framework for manipulating different teaching strategies. The proof and tutoring tasks are parallel. Separating their dialogue management makes the tutoring task reusable for different problems and domains.

We now define the task dialogue moves. For the purposes of this thesis we talk about student-task vs. tutor-task dialogue moves.

### 4.4.1 Student-task Dialogue Moves

The following student-task dialogue moves constitute special requests from the student. The analysis provided here is necessary for the identification and classification of such dialogue moves, in order to be able to provide tutorial feedback for them. In performing such dialogue moves, the student takes the dialogue initiative. However, the tutor is still responsible for the task initiative to be in a position to give feedback. Therefore, there need to be two levels of dialogue management to capture this two forms of initiative.

#### 4.4.1.1 Request-assistance

An utterance with which the speaker requests assistance with the task. It concerns only specific information requests, e.g., about concepts.

**Specifications** It is a superclass with values all kinds of assistance that can be dealt with, namely the domain information captured in passive hint categories. It also comprises two more categories: one representing that the requested assistance is irrelevant, and one representing that it belongs to a different theory.

**Obligations** The tutor is obliged to address it, but will really only *answer request-assistance* based on teaching model considerations.

**NL Examples**

(4.13) **S1:** "Ich kann nicht anfangen."
*[I can't start]* (soc5k)

(4.14) **S:** "What is a P?" (constructed example)

**Relation to moves in other dimensions** The forward-looking function is *info-request*, *statement*, or *action-directive*.

**Notes** This move is analysed for the possible aspects in the domain in Chapter 5.

#### 4.4.1.2 Resign

An utterance with which students indicate that they are giving up the proof-task, as opposed to asking for specific help.

**Specifications** No further specifications are required or provided by `Menon`.

**Obligations** The tutor has to address it at least. The rest of the behaviour is dependent on the teaching model.

**NL Examples**

(4.15) **S3:** "Gebe auf."
*[I give up.]* (soc5p)

(4.16) **S7:** "Komme nicht weiter."
*[Can't move on.]* (did15p)

(4.17) **S5:** "Ich moechte die antwort wissen."
*[I want to know the answer.]* (soc5k)

(4.18) **S2:** "Wenn ich das wuesste!"
*[If only I knew that!]* (soc20k)

**Relation to other moves** It is different from *request-assistance*, because the student states that they are giving up, one way or the other.

"I don't know" is a *reject* (and *statement*) in DAMSL, not an *answer*, whereas in [Core *et al.*, 2002] it is an *answer*, because according to the speaker the question is being answered. We understand an *answer* as resolving a question and an *address-info-request* as just addressing it. So, "I don't know" effect utterances are not treated as *answers*, but rather as *resigns* and *address-info-requests* at the BACKWARD-LOOKING level.

#### 4.4.1.3 Request-evaluation

An utterance with which students inquire explicitly about their progress.

**Specifications** No further specifications are required by `Menon`.

**Obligations** The tutor has to address it and probably answer it, depending on the teaching strategy, by either an *encourage* ("You are doing fine"), a *domain-contribution-evaluation* ("Your answer is mainly correct, but ...") or a combination of the two ("You are doing fine, but you have to concentrate on the ...").

**NL Examples (constructed)**

(4.19) **S:** "How am I doing?". This is requesting an evaluation of the overall performance.

(4.20) **S:** "Was that right?". This is requesting an evaluation of the current student answer.

#### 4.4.1.4 Time-out

A period of time where the student remains idle.

**Specifications** No further specifications are required by `Menon`.

**Obligations** The tutor has to address it and try to find out the reason for the idleness.

**Notes** This is not strictly speaking a dialogue move. For the formalisation of the task dimension, though, it transpired that it functions rather as a dialogue move, as it requires general pedagogical feedback.

### 4.4.2 Tutor-task dialogue moves

**Proof Task** It is concerned with resolving the domain task, in our domain finding a mathematical proof to a problem. It comprises the following dialogue moves.

#### 4.4.2.1 Domain-contribution

An utterance that attempts to directly resolve the proof task or a related issue to the best of the speaker's capacity, even if wrong or irrelevant.

**Specifications** It always takes as a parameter the domain-contribution category assigned to it. This can differ according to the teaching model. We analyse the categories assumed in our model in Chapter 5, Section 5.3.3.

**Obligations** The tutor is obliged to *acknowledge* it and also perform one of the *domain-contribution-evaluation* dialogue moves.

**NL Examples**

(4.21) **S2:** "Wegen A $\subseteq$ K ( B ) = A $\cap$ B = $\emptyset$ (obiger Schritt) und analog B $\subseteq$ K ( A ) = B $\cap$ A = $\emptyset$ und A $\cap$ B = $\emptyset$ = B $\cap$ A (offensichtlich) folgt die Behauptung A $\subseteq$ K ( B ) = B $\subseteq$ K ( A )"
*[Because $A \subseteq K ( B ) = A \cap B = \emptyset$ (above step) and analogously $B \subseteq K ( A ) = B \cap A = \emptyset$ und $A \cap B = \emptyset = B \cap A$ (obviously) follows the hypothesis that $A \subseteq K ( B ) = B \subseteq K ( A )$ ]*

(4.22) **T:** "Is A a subset of B?"
**S:** "Yes" (= A is a subset of B) (constructed example)

**Contrasting examples**

(4.23) **S5:** "Habe Probleme mit Potenzmenge..."
*[I'm having problems with the powerset...]* (did15p). This is not a *domain-contribution* although it mentions powerset. It can be handled by the dialogue manager without consulting the proof manager.

(4.24) **S:** "What is a K(A)?" (constructed example) This is not a *domain-contribution*, but a *request-assistance*.

(4.25) **T2:** "Das ist nicht richtig! Sie müssen als erstes die wenn-dann-Beziehung betrachten."
*[This is not correct. First, you have to deal with the if-then relation.]* (soc23k).
This is a *hint*, as the tutor is not advancing the proof task to the best of her ability. That would mean giving away the whole proof. She is, however, contributing to the tutoring task by giving this piece of information and hoping to effect learning.

**Notes** The tutor does not contribute to the proof task, but to the tutoring task.

*Domain-contribution* may contain domain knowledge, but this is not a decisive factor either way. On the contrary, any utterance from the student that does not contain domain knowledge, but needs to be evaluated (e.g., Example 4.22) is a *domain-contribution* and requires linguistic interpretation for what counts as an *answer* to it (in the example "yes", "no").

**Tutoring Task** It includes dialogue moves that do not try to resolve the task directly, as in the PROOF TASK subdimension, but rather aim at assist the student resolve the task. It comprises the following dialogue moves.

#### 4.4.2.2 Check-origin-problem

A question that the tutor asks in order to find out if and what problem the student has. It does not advance the proof task.

**Specifications** It may take as parameters either the student-task dialogue move that caused it, or the domain-contribution category if it was caused by one.

**Obligations** This move discharges the obligation of the tutor to address a preceding question or assertion by the student. The student has to address it. This is taken care of by the FORWARD-LOOKING level.

**NL Examples**

(4.26) **T10:** "Koennen Sie das noch genauer erklaeren?"
*[Can you explain that in more detail?]* (soc20k)

(4.27) **T:** "What is it that you do not understand?"(constructed example)

#### 4.4.2.3 Align

An utterance with which the person who knows more (the tutor) tries to get evidence that the interlocutor shares the same knowledge, as the speaker believes is the case.

**Specifications** It takes as a parameter a hint, the content of which is the target of the *align*. That is, the tutor is trying to make sure that the student understands this content.

**Obligations** An *align* imposes the obligation on the student to answer it, but a simple "yes" does not discharge the obligation to answer it. This is taken care of at the FORWARD-LOOKING function, where it is a *diagnostic-query*.

**NL Examples**

(4.28) **T6:** "...Warum haben wir zuerst die wenn-dann Beziehung betrachtet?"
*[Why did we first deal with the if-then relation]* (soc21k).
This is an align when the tutor is trying to re-elicit that information after having given it once.

**Relation to moves in other dimensions** In the FORWARD-LOOKING dimension *align* is probably always a *diagnostic-query*.

#### 4.4.2.4 Domain-contribution-evaluation

An utterance that provides an evaluation of the student's performance by signalling each of the *domain-contribution* categories. As such, it is a supercategory and its subcategories comprise signalling the particular domain-contribution categories.

**Specifications** This is a superclass that comprises one subcategory for every defined domain-contribution category in the model. We identify the domain-contribution categories: signal-accept, signal-irrelevant, signal-step-size, signal-misconception, signal-ill-formed, signal-complete-inaccurate, signal-partial-answer, signal-complete-partially-accurate, signal-near-miss, signal-unknown, signal-missing-basic-knowledge, signal-wrong-linguistic-term, signal-wrong, signal-other. When it is produced after a *request-evaluation*, it takes this dialogue move as a parameter. This makes possible a different NL realisation if necessary. For example, a *domain-contribution-evaluation* can be implicit, but when it is specified by the parameter *request-evaluation* then it should rather be explicit and more verbose.

**Obligations** It discharges the obligation to answer a *request-evaluation*.

**NL Examples (constructed)** *Signal-near-miss* can be implicit and taken care of by the hint that follows it.

**NL Examples**

(4.29) **T:** "Is this really what you wanted to say?"
This is a `discrepancy` hint and at the same time an implicit *signal-near-miss*.

(4.30) **T:** "There's a minor problem with your answer."
This is an explicit *signal-near-miss*.

(4.31) **T:** "What you said is very close to the correct answer, apart from a minor problem."
This is an explicit *signal-near-miss* with *request-evaluation* as a parameter.

**Notes** This category comprises dialogue moves that are the TASK dimension counterpart of the BACKWARD-LOOKING moves *accept*, *accept-part*, *reject* and *reject-part*. *Domain-contribution-evaluation* is produced regularly, to let students know how their contribution was. However, students might also ask explicitly for an evaluation.

#### 4.4.2.5 Encourage

The speaker (tutor) gives some positive feedback to the hearer (student) in order to encourage her to continue. It does not presuppose that the speaker accepts the domain contribution as in 4.33, but it might, as in 4.32, where the adverb "vollkommen" (*absolutely*) indicates that it is an *encourage*.

**Specifications** If produced after a *request-evaluation* it takes this as parameter.

**Obligations** It discharges the obligation to address a *resign*.

**NL Examples**

(4.32) **T6:** "Das ist vollkommen richtig!" (soc17d)
*[This is absolutely right!]*

(4.33) **T9:** "Sehr gut!"
*[Very well!]* (soc20k)

(4.34) **T4:** "Der ansatz ist richtig."
*[The beginning is correct]* (soc2k)

**NL Examples (constructed)**

(4.35) **T:** "Good."
This is an *encourage* without any parameter.

(4.36) **T:** "It's not easy, but you'll get there!"
This is an *encourage* without parameter *request-evaluation*.

**Notes** As with *domain-contribution-evaluation*, when an *encourage* is produced after a *request-evaluation*, it takes the *request-evaluation* as a parameter for adapting the NL realisation of it.

#### 4.4.2.6 Prompt

An utterance that requests the hearer explicitly to proceed with the task and provide further information.

**Specifications** Depending on what exactly the student is being prompted for, it takes one of the parameters prompt-action, prompt-step, or prompt-resign. The last captures prompting the student after a *resign* to try harder.

**Obligations** The student is obliged to address it by trying to proceed with the task.

**Relation to moves in other dimensions** It can be realised as a *diagnostic-query* or an *info-request* in the FORWARD-LOOKING dimension (e.g., Example 4.37).

**NL Examples**

(4.37) **T3:** "Wie geht es weiter?"
*[How does one continue?]* (did15d)

(4.38) **T2:** "Wie koennte das gehen?"
*[What could the answer be?]* (did16k)

(4.39) **T4:** "Wie koennte es damit weitergehen?"
*[How could one continue with this?]* (did15k)

**Notes** It is usually the tutor who would utter *prompts*. For the student such an utterance will be a *resign* at the task level and *info-request* forward-looking.

#### 4.4.2.7 Hint

A hint can take the form of eliciting information that students are unable to access without the aid of hints, or information whose relevance to the problem at hand is not clear to the students. Alternatively, a hint can point to an inference that students are expected to make based on knowledge available to them, which helps the general reasoning needed to deal with a problem [Hume *et al.*, 1996b].

The initiation of hints can be due to various reasons:

(i) the speaker (tutor) observes that the other interlocutor (student) is not making any progress in the task

(ii) the interlocutor (student) asks a question and the speaker (tutor) does not want to answer it directly

(iii) the interlocutor (student) gives the wrong answer or asks the wrong question in response to a speaker's (tutor) question

**Obligations** Partial answers from the tutor in the form of *hint* discharge the obligation to address the student's questions or utterances. There are examples in the BE&E corpus where the tutor explicitly states her method or the student shows she is aware of it, such as "Very good. You answered your own question" or "I'll give you another hint.", from the tutor and "I need another hint", from the student. Statements like these constitute additional support for the obligations involved. They show the dialogue participants' awareness of their respective roles in this special social context [Tsovaltzi and Matheson, 2002; Hulstijn, 2003].

In the next section, we scrutinise the dialogue move *hint*, we provide specifications for every hint category defined and extensive examples. In Section 4.8, we discuss the relation of *hint* to other moves in the dialogue-move taxonomy.

## 4.5 Hint Taxonomy

We now explore hints and show how their definition makes further use of our instructional points and pedagogical model. In order to capture the different underlying cognitive functions of a hint in our hint taxonomy, we define hint categories across different dimensions. The cognitive functions of a hint can be common for different surface realisations, therefore we define hint categories within the top level dialogue-move taxonomy.

### 4.5.1 Motivation and Structure of the Hint Taxonomy

We define four dimensions of hints:

1. The *domain knowledge* dimension captures the needs of the domain, distinguishing different instructional points for skill acquisition in problem solving.
2. The *elicitation status* dimension distinguishes between the information being elicited and degrees in which it is provided.
3. The *problem referential perspective* dimension distinguishes between views on discovering an inference. It includes the conceptual perspective, which points to an inference by referring to relations between pieces of domain information, and the pragmatic perspective, which addresses pragmatic aspects of pointing to an inference like the number of parts expected for an answer to be complete.
4. The *inferential role* dimension captures the type of inference concerning the instructional point being addressed. For conceptual hints, we distinguish between whether what is addressed is the inference per se, or some control on top of it, its meta-reasoning. For example, the rule of inference can be elicited directly, or the reasoning for finding the right rule of inference can be elicited. For pragmatic hints, the inferential role refers to the applicable pragmatic reasoning, for example whether pragmatic elements of the student's answer are addressed (`speak-to-answer`), or whether the student is pointed to some relevant infrormation (`point-to-information`).

A hint category is determined by the combination of these four dimensions (cf. Figure 4.1). That is, one category is a point in the space defined by the four dimensions, where the point itself is defined by the choices made on each of the dimensions. Different combinations are potentially useful, for alternative teaching models.

We shall first describe the four dimensions in detail and then give example hint categories, as defined for our teaching model.

#### 4.5.1.1 The Domain-Knowledge Dimension

In Chapter 2 we discussed the pedagogical and cognitive relevance of instructional points and in Chapter 3 we defined the different instructional points that

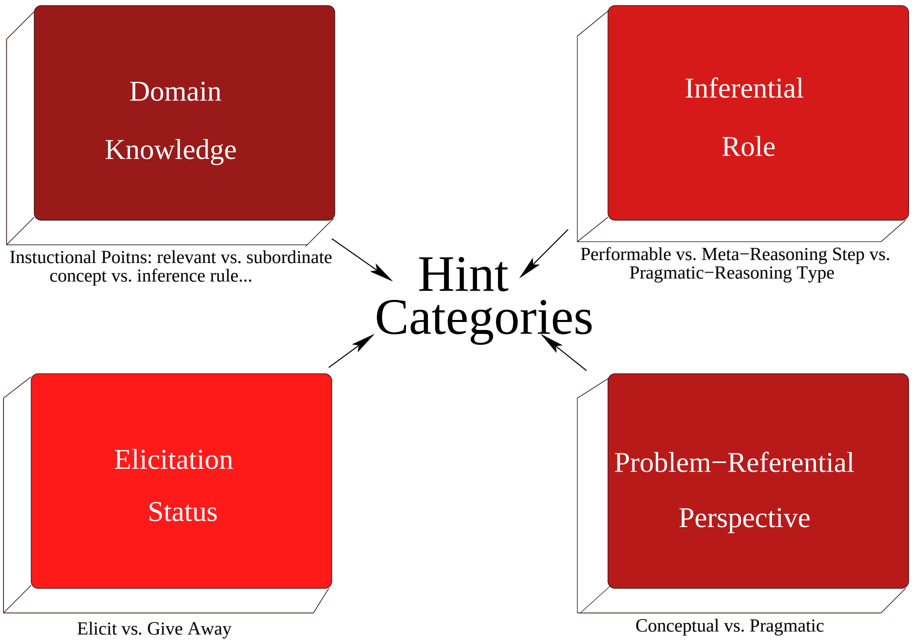


Figure 4.1: Definition of Hint Categories via Hint dimensions

make up this dimension, in terms of the domain knowledge they refer to. Here, we look at the pedagogical and cognitive relevance of the instructional points and at their use for the definition of hint categories.

In this dimension, we classify instructional points under five classes, based on a *subordination relation*, that is their order with respect to the amount of information they reveal. The classes are:

1. `Domain relation`
2. `Domain object`
3. `Rule of inference`
4. `Substitution`
5. `Proof step`

These classes correspond to the categories of instructional points defined in Chapter 3. The domain knowledge captured in them, that is the concepts and relations that we define, are used as possible hint aspects for each of the above classes. In other words, they constitute one decision point in the hint taxonomy, which represents the domain knowledge to be addressed by the hint.

#### 4.5.1.2 The Elicitation-Status Dimension

This dimension draws a distinction between the active and passive function of hints. The *active* function of hints looks forward and seeks to help students in accessing further information that will bring them closer to the solution by means of eliciting. Students have to think of and produce the answer that is hinted at. Active hints enable the creation of personalised schemata. They are a means to avoid providing specific declarative knowledge, but guide students to form their own knowledge, and hence a schema that they can re-apply [Price *et al.*, 1997].

The *passive* function of hints refers to the piece of declarative information that is provided each time in order to bring the student closer to some answer. The tutor gives away some information, which might have been previously elicited without success. This captures the basic idea of scaffolding, i.e., of providing the knowledge that the learner does not possess while requiring the use of already possessed knowledge [Wood *et al.*, 1976].

#### 4.5.1.3 The Problem-Referential-Perspective Dimension

This dimension distinguishes between two modes of referring to the instructional points; conceptual and pragmatic. `Conceptual` hints directly refer to instructional points using domain information. In our domain, they make use of mathematical concepts or reasoning, like the premise and the conclusion, the rule of inference, the proof step. `Pragmatic` hints, in contrast, are opposed to the analytic and deductive way of thinking that is reflected in the conceptual hints. For instance, a pragmatic hint may inform the student of the number of subparts that are necessary for an answer to be complete, or point the student to a previous occurance of some domain information in the course of the same tutoring session, or correct some mistake in terminology.

The pragmatic class takes the cognitive state of students into account and aims at increasing the students' motivation. For instance, the tutor may think that the student has a basic understanding of the conceptual aspect and is not far from the desired answer, therefore the pragmatic information is enough. Pragmatic hints, in effect, only point to conceptual information that is implicit (not available to consciousness or verbalisable) and cannot be taught as such. It takes advantage of patterns that do not themselves constitute instructional points, but help to recognise them. This in turn assists the building or selection of the appropriate schema [Price *et al.*, 1997].

#### 4.5.1.4 The Inferential-Role Dimension

This dimension captures the kind of inference that is addressed by the hint. For conceptual hints, we distinguish between *performable-step* and *meta-reasoning* instructional points. *Performable-step* instructional points have to be explicitly represented in the proof step. *Meta-reasoning* instructional points help derive the performable step, but are not necessarily represented in the proof step.

**Performable Steps** *Performable steps* are the steps that can be found in the proof. These include premises, conclusion and inference methods such as lemmata, theorems, definitions of concepts etc.

Performable-step hints take care of declarative knowledge necessary for eliminating errors. Their active function aims in a sense at forcing students to pay attention to such aspects of the proof that are relevant to identifying the schema to be applied. These aspects are normally pre-consciously recognised if the schema already exists [Price *et al.*, 1997].

**Meta-reasoning Steps** *Meta-reasoning steps* consist of everything that leads to the performable step, but cannot be found in the proof as such. To be more specific, meta-reasoning consists of everything that could potentially be applied to proofs in general. It consists of everything that explains the performable step, building the motivation for the instructional points. It involves general proving techniques and methodology, which are theory independent. As soon as a general technique is instantiated for the particular proof, it belongs to the performable step level. For example, although the process of substituting results in a performable step, the process itself is meta-reasoning.

The meta-reasoning could be abstracted from performable-step hints in the form of schemata built by students and suiting their cognitive state. If students are not capable of this abstraction, meta-reasoning hints help them do so by motivating the performable-step instructional points and thus providing additional help. Such rule-based instruction has been found to be inappropriate on its own, as it hinders the acquisition of personalised schemata [Price *et al.*, 1997]. However, in combination with performable-step hints, they elevate cognitive load, motivate the student and reinforce instructional points. In particular, active meta-reasoning hints (cf. Elicitation Status Dimension) are pedagogically speaking appropriate for students who already have some *schema*, but get stuck in applying it, as our experiments have shown (cf. Chapter 1). Those subjects seemed to be hampered by performable-step information hints.

Passive meta-reasoning hints subsume the corresponding passive performable-step hints, that is, they include their information. Furthermore, meta-reasoning subclasses capture the class subordination in so far as they motivate the domain hints, which themselves follow the subordination.

For pragmatic hints the inferential role corresponds to the way hints can help with inferring the domain knowledge by reference to pragmatic aspects of it (cf. Problem Referential Dimension). We identify three classes:

1. speak-to-answer, which addresses the student's *domain-contribution* directly

2. point-to-lesson, which addresses the declarative knowledge that has been presented to the student in the lesson material

3. take-for-granted, which refers to information that the student is not required to know, as the current session does not deal with it

1. DOMAIN KNOWLEDGE: *premise-conclusion* $\longrightarrow$ the current premise and conclusion
2. INFERENTIAL ROLE: *meta-reasoning*
3. ELICITATION STATUS: *give away information*
4. PROBLEM-REFERENTIAL PERSPECTIVE: *conceptual*

Figure 4.2: Example of hint category with its values in the four dimensions.

### 4.5.2 Hint Categories

We now examine the hint categories that serve our pedagogical goals and also help `Menon` to provide as much specification as possible for the NL realisation.

Based on our tutoring model in Chapter 2, the hint categories we define are, in general, pointers to the instructional points from different perspectives, but always give non-redundant information, as cognitive load theory advises [Sweller, 1988]. Moreover, they do not include explicit mention of a schema, but only suggest some basic heuristics, as the former would mean imposing a particular schema. This in turn inflicts unnecessary cognitive load and disallows learning based on the existing cognitive structures as well as implicit learning [Stanley *et al.*, 1989]. In that spirit, hints do not make the relation between instructional points explicit, and the instructional points defined in Chapter 3 were conceived with the aim of keeping them abstract enough to allow the schema acquisition [Sweller, 1988; 1989]. At the same time, the realisation of the hints does not present the definition of the instructional point itself, but its instantiation for the particular proof and proof step, that is, the problem at hand. This is also crucial for the full automation of NL hints, as instructional points can be instantiated for any proof currently constituting the task.

Figure 4.2 shows and example of how the decisions in the hint dimensions define a hint category. The task assumed is to prove that *if* $A \subseteq K(B)$*, then* $B \subseteq K(A)$. The hint category is `give-away-premise-conclusion`. Given the values in the four dimensions, a realisation that could be produced by the NL generator for this hint would be: "What we want to prove here is that *if* $A \subseteq K(B)$*, then* $B \subseteq K(A)$ holds, by proving $B \subseteq K(A)$ under the assumption that $A \subseteq K(B)$".

In the following, we provide examples of possible surface realisations of hints. We keep the phrasing in examples of one hint category the same and provide templates for NL formulation, to allow the comparison between the same hint category for instantiation of different instructional points. For an analysis of linguistic phenomena in tutorial dialogue and a computational model for surface realisation of tutorial feedback see, for example, [Porayska-Pomsta and Pain, 2000]. To illustrate some assumptions made in our NL realisations and the possibilities that our approach provides for generating contextual realisations of hints, here are some constructed examples of different possible realisations depending on dialogue context.

The hint `give-away-relevant-concept` can be realised in many ways, for instance:

(4.40) ...
**S:** (the student is silent for some time)
**T:** "You can start by considering the < Relevant Concept >." (*statement, open-option*)

(4.41) ...
**S:** "What shall I do now?" (*info-request, request-assistance*)
**T:** "Start by considering the < Relevant Concept >." (*action-directive, address-question*)

(4.42) ...
**S:** "I want to know the answer." (*statement, resign*)
**T:** "Why don't you try first to consider the < Relevant Concept >." (*open-option, address-statement*)

In Example 4.40, the tutor reads in the student's silence that she might not know how to continue. Hence, she cautiously states a possibility to proceed. In Example 4.41, the student indicates that she is at a loss, so the tutor just instructs her to use the Relevant Concept without further ado to avoid more confusion. In Example 4.42, the student does not know how to proceed and does not want to try either. The tutor is not prepared to give the answer away, for pedagogical reasons, but she has to address the student's request. So, she informs the student how she can carry on trying, but addresses her wish to resign. Clearly, such NL realisations may also depend on student model considerations and are produced in varied discourse contexts (here for simplicity this consists of the previous student turn only) and intend different communicative goals, which the tutor wants to achieve on top of presenting the underlying cognitive content. However, all examples realise the same pedagogical feedback (here they realise the hint `give-away-relevant-concept`). Such nuances in realisation play an important role in what makes human tutors effective [Moore, 1993; Porayska-Pomsta and Pain, 2000; DiPaolo *et al.*, 2004; Freedman, 2000]. The approach to automating feedback presented here takes that into account and allows the possibility of varying the NL realisations to cater for those nuances.

#### 4.5.2.1 DPC Hints

The hints categories defined here combine the aspects Domain Information, Performable Step, and Conceptual. We present the active and the passive equivalent of the hints.

**4.5.2.1.1 Domain-Relation Hints** The domain-relation class belongs to the meta-reasoning (cf. Section 4.5.2.2) and has no function in the performable step.

**4.5.2.1.2 Domain-Object Hints** `Domain-object` hints address an object in the domain.

`Elicit-relevant-concept` It asks the student for the most prominent concept in the proposition or formula under consideration. This might be, for instance, the concept whose definition the student needs to use in order to proceed with the proof, or the concept that will in general lead students to understand which Rule of Inference they have to apply.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "Look for the thing in the expression with which you can start working." or
"What is the thing in the expression from which you can start working?"

`Give-away-relevant-concept` It gives the relevant concept away and asks the student to consider the Relevant Concept for the proof step.

**Specifications** Relevant Concept[3].

**NL Examples** "You can start by considering the $<$ Relevant Concept $>$ (e.g., powerset)."[4].

**Notes** For a definition of Relevant Concept based on the Rule of Inference, see Chapter 3.

`Elicit-subordinate-concept` It asks the student for the Subordinate Concept in connection to the Relevant Concept.

**Specifications** Relevant Concept.

**NL Examples** "Think of what you know about the $<$ Relevant Concept $>$ (e.g., powerset) that can help you manipulate the expression." or
"Is there anything that connects to the $<$ Relevant Concept $>$ (e.g., powerset) and can help you manipulate the expression ?"

**Notes** It is provided when the Relevant Concept is known.

[3]The specifications refer to the definitions in Chapter 3 and to Appendix B.

[4]Where appropriate in the NL examples, we keep them generic, providing the hint specifications to show how these specifications would be used for the automatic realisation of the hints. We also give an example for the possible instantiation or realisation of the specifications. We employ a categorisation based on the Domain Techniques defined in Chapter 3. A more extensive list of NL examples is available in Appendix D.

<u>`Give-away-subordinate-concept`</u> It gives the subordinate concept away.

**Specifications** Subordinate Concept.

**NL Examples** "Think of the $<$ Subordinate Concept $>$ (e.g., subset)."

**Notes** For a definition of the Subordinate Concept based on the Rule of Inference, see Chapter 3.

The passive function of domain-object hints is used to elicit the applicable Rule of Inference, and, therefore, constitutes the active function of the respective class.

**4.5.2.1.3 Inference-Rule Hints** The instructional point Rule of Inference should not be confused with the rule of inferences of logic, as it does not share all their attributes. So, although statically extraction and insertion are the same, the instantiation of them for the instructional point Rule of Inference is different, because direction plays a role. That means that pedagogically they are different rules.

<u>`Elicit-inference-rule`</u> It asks the student to name the rule that needs to be applied.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "What rule can you use here?"

**Notes** `Elicit-inference-rule` is used to elicit the Substitution of the rule for the problem at hand.

<u>`Give-away-inference-rule`</u> It names the Rule of Inference to be used.

**Specifications** Rule of Inference, Relevant Concept.

**NL Examples**

NL Template: "You have to use $<$ Rule of Inference $>$ of $<$ Relevant Concept $>$."

- Occurrence state[5] of definitions or substitutions: "You have to use the definition of powerset."
- Case distinction: "You have to use the cases deriving from the disjunctive definition of the *union* $\cup$.", where *the union* $\cup$ is a disjunctively defined concept, defined as $U \cup V = \{x|x \in U \ or \ x \in V\}$ and the cases deriving from its definition would be $x \in U$ or $x \in V$.

[5] For the definition of occurrence state see Applendix A, Section A.2.4.1

- Induction: "You have to use the steps deriving from the inductive definition of *the set of all finite subsets of X.*", where *the set of all finite subsets of X* is the inductively defined concept, and $P_f(X)$ *is the set of all finite subsets of X* if: (i) $\emptyset \in P_f(X)$ (ii) $A \in P_f(X), x \in X \Rightarrow A \cup \{x\} \in P_f(X)$.

- Occurrence state of quantifier: "You have to use the rule elimination of the universal quantifier for the *for all.*", where *for all* is a realisation for the Relevant Concept, which is the quantifier type *for all* or *there exists.*

- Occurrence state of connectives: "You have to use the rule elimination of the equivalence for the equivalence.", where *equivalence* is a realisation for the Relevant Concept, which is the connective type *equivalence*, *equality*, *implication*, *conjunction*, or *disjunction.*

**Notes** The Rule of Inference for case distinction and induction, includes the cases and the inductive steps, respectively.

`Elicit-basic-knowledge` It asks the student for the Basic Knowledge, that is some declarative knowledge, which the tutor has become aware that the student was missing. That knowledge can be definitions of concepts, definition of Rule of Inferences etc. (cf. Chapter 5).

**Specifications** Basic Knowledge Reference.

**NL Examples** "What is the < Basic Knowledge Reference > (e.g., the definition of powerset, or implication elimination etc.)?"

**Notes** Basic Knowledge is not strictly speaking an instructional point, but it is a domain mistake that is related to this class in the domain knowledge dimension, as it refers to definitions of either rules or Domain Techniques (cf. Chapter 5). The Basic Knowledge elicits, in effect, the Rule of Inference, as all hints do before the Rule of Inference is known. Therefore, the corresponding class subtask (cf. Chapter 6) deals also with Basic Knowledge.

`Give-away-basic-knowledge` It gives away the information on the defined constants that the student does not know.

**Specifications** Basic Knowledge Reference, Basic Knowledge, Relevant Concept.

**NL Examples**

NL Template: "The rule for the < Basic Knowledge Reference > is: < Basic Knowledge >"

- Occurrence state of definitions or substitutions: "The rule for the definition of powerset is: $P(V) = \{U | U \subseteq V\}$."

- Case distinction: "The rule for the case distinction of the *union* $\cup$ is: for the expression to hold for the union of $U$ and $V$ it has to hold for the different cases deriving from its disjunctive definition $x \in U$ or $x \in V$."
- Induction: "The rule for the induction of *the set of all finite subsets of* $X$ is: for the expression to hold for *the set of all finite subsets of* $X$, it has to hold for the different steps deriving from its definition $\emptyset \in P_f(X)$ and $A \in P_f(X), x \in X \Rightarrow A \cup \{x\} \in P_f(X)$.
- Occurrence state of quantifiers: "The rule for the elimination of the *for all* is: for an expression to hold for all x, it has to hold for an arbitrary but fixed x."
- Occurrence state of connectives: "The rule for the elimination of equivalence $\Leftrightarrow$ is: an expression with an equivalence holds if it holds from left to right and from right to left".

**4.5.2.1.4 Substitution Hints** Performable-step Substitution hints deal with how the Substitution should be done. The result of the substitution is the proof step and is dealt with by the proof-step hints below. (cf. Section 4.5.2.2).

`Elicit-substitution` It asks the student to apply the Rule of Inference to the given expression, that is to bind the variables in it.

**Specifications** Rule of Inference.

**NL Examples** "Try now to apply the rule $<$ Rule of Inference $>$ (e.g., the definition of powerset) to the expression."

`Give-away-substitution` It explains to the student the way the substitution is done. Any wrong substitution (i.e., of the source, target or Rule of Inference) is always nonetheless a substitution and treated here.

**Specifications** Substitution.

**NL Examples** "You can substitute the appropriate parts of the expression you are dealing with for the variables in the rule you are applying."

**Notes** References to substitution of variables appeared in our corpus. e.g., "That is not totally right. The element of which set do you have to handle?" (soc2K, T3), where the student correctly wanted to handle the problem by using the element of a set, but she talked about the element of A instead of B, which was the correct set.

This hint gives general instructions on substituting. Problems with substituting specific parts are treated by `elicit-` and `give-away-ill-formed`, as well as by the domain-relation hints. We leave it for future work to guide the student even more closely through the substitution, as for the moment there is no evidence that it is necessary, but it would be an expensive task.

`Elicit-ill-formed` It informs students that their formulation is not mathematically correct, and asks them to correct it. It thus elicits the correct Substitution.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "What you wrote there is syntactically not completely correct. Can you correct it?"

**Notes** Ill-formed is a domain mistake that relates to the Substitution.

`Give-away-ill-formed` It lets the student know what is wrong in their formulation, without giving away the whole Substitution.

**Specifications** Correct Formulation.

**NL Examples** "The right way to write this is $<$ Correct Formulation $>$ (e.g. $P(C \cup (A \cap B))$ instead of $PC \cup (A \cap B)$."

#### 4.5.2.1.5 Proof-Step Hints

`Elicit-proof-step` It asks the student to write the complete proof step.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "Can you now write the whole step?"

`Give-away-proof-step` It gives away the whole proof step.

**Specifications** Proof Step.

**NL Examples** So, this step is $<$ Proof Step $>$ (e.g. Let $A \subseteq K(B)$, we will prove that $B \subseteq K(A)$)."

### 4.5.2.2 DMRC

The hints categories defined here combine the aspects Domain Information, Meta-Reasoning Step, and Conceptual. We present the active and the passive equivalent of the hints.

Meta-reasoning is used to motivate the performable step and what is present in it. For example, `relevant-concept` and `subordinate-con- cept` should be accompanied by their equivalent meta-reasoning. In the case of domain relation concepts though (antithesis, duality etc.) any meta-reasoning is captured in the hint itself. Namely, that the concept chosen by the student stands in such a relation to the desired concept.

The meta-reasoning that we provide is informed by the principle of "local axiomatics". For instance, we provide information on the techniques to be used (e.g., universal quantifier elimination), but do not enter the logic level of explaining the techniques themselves (e.g., why it suffices to prove an expression that involves a universal quantifier for an arbitrary constant). Our aim is rather that the student understands when and how to apply the techniques.

For the definition of most hints in this dimension we made use of the didactic condition feedback, which consisted always of a passive hint plus its motivations, as our wizard-tutor was instructed to do.

**4.5.2.2.1 Domain-Relation Meta-Reasoning Hints** All hints that correspond to the domain relations defined in the taxonomy belong here. We do not make use of their active function, as it is not part of the tutorial goal. The active function would be applicable if we were really concentrating on teaching concepts (declarative knowledge), or making the student aware of the domain hierarchy.

For all passive domain relation hints, the fact that they bear the specific relation to the required concept is also the content of the realisation, as for the NL example for antithesis and hypotaxis below. The relations are defined in Appendix A.

So we identify the following hints:

`Give-away-antithesis` It informs the student that the concept used is in antithesis to the one expected.

`Give-away-duality` It informs the student that the concept used is in duality to the one expected.

`Give-away-converse` It informs the student that the concept used is the converse to the one expected.

`Give-away-hypotaxis` It informs the student that the concept used is in hypotaxis to the one expected.

`Give-away-specialisation` It informs the student that the concept used is a specialisation of the one expected.

`Give-away-generalisation` It informs the student that the concept used is a generalisation of the one expected.

`Give-away-primitive` It informs the student that the concept used is primitive to the one expected.

**Specifications** Domain-Relation Name, Domain-Relation Concept.

**NL Examples**

- Antithesis NL Template: “The $<$ Domain-Relation Concept $>$ is the $<$ Domain-Relation Name $>$ of the concept you need here.”
  Antithesis NL Example: “The $\in$ is the opposite of the concept you need here.”’, where the concept needed is the $\notin$.

- Hypotaxis NL Template: "The < Domain-Relation Concept > is the < Domain-Relation Name > of the concept you need here."
  Hypotaxis NL Example: "The $\in$ is the hypotaxon of the concept you need here.", where the concept needed is the *subset* $\subseteq$.

**Notes** In the examples above, "opposite" is the NL realisation for the Domain-Relation Name *antithesis*, and "part of the definition" for *hypotaxon*. The realisations for the rest of the domain relations would be equivalent, but with a different NL description based on the domain relation they handle.

#### 4.5.2.2.2 Domain-Object Meta-Reasoning Hints

`Elicit-relevant-concept-meta-reasoning` It elicits the reasoning for finding the Relevant Concept as well as the role it plays for finding the next proof step in general.

**Specifications** Specific Method.

**NL Examples**

- Forward: "Now, you should look for something in the problem that would help you < Specific Method > (e.g., manipulate the expression towards what you are trying to prove)."
- Backward: "Now, you should look for something in the problem that will help you < Specific Method > (e.g., simplify what you are trying to prove)."

**Notes** To be produced when the Subordinate Concept is not known.

The Specific Method can be used to differentiate between NL realisations for forward vs. backward steps, as in the NL Examples above.

`Give-away-relevant-concept-meta-reasoning` It gives away the Relevant Concept and the meta-reasoning for it.

**Specifications** Specific Method, Relevant Concept.

**NL Examples**

NL Template: "We start with the < Relevant Concept > because it's central in the problem and can help you < Specific Method >."

- Forward: "We start with the powerset, because it's central in the problem and can help you manipulate what you want to prove."
- Backward: "We start with the *if-then relation*, because it's central in the problem and can help you simplify it."

**Notes** To be produced when the Subordinate Concept is not known.

Examples from our corpus are:

- "We deal first with the *if-then* relation to simplify the whole expression" (soc23k, T8), where the Relevant Concept is the *if-then* relation (the implication).
- "…You only have to consider that both the terms in $(P(A) \cup P(C)) \cap (P(B) \cup P(C))$ are connected by intersection" (soc17p, T4), where intersection is the Relevant Concept.

`Elicit-relevant-concept-meta-reasoning(subordinate-concept)` It asks the student for the Relevant Concept by explaining its relation to the Subordinate Concept.

**Specifications** Subordinate Concept.

**NL Examples**

NL Template: "Now, find something in the expression, which you can connect to the $<$ Subordinate Concept $>$ and can help you <Specific Method>."

"Now, find something in the expression, which you can connect to the subset and can help you find the right rule for the next step.", where the Relevant Concept is the powerset, the Inference Rule the definition of powerset, and the step is forward.

**Notes** To be produced when the Subordinate Concept is known.

`Give-away-relevant-concept-meta-reasoning(subordinate-concept)` It gives away the Relevant Concept and explains its use in connection to the Subordinate Concept.

**Specifications** Specific Method, Relevant Concept, Subordinate Concept.

**NL Examples**

NL Template: "You can consider the $<$ Relevant Concept $>$, because it is connected to $<$ Subordinate Concept $>$ and, therefore, you can use it to $<$ Specific Method $>$."

"You can consider the powerset, because it is connected to the subset and, therefore, you can use it to manipulate the expression.", where the step is forward.

**Notes** To be produced when the Subordinate Concept is known.

`Elicit-subordinate-concept-meta-reasoning` It asks for the Subordinate Concept in connection to the relevant concept by explaining its function.

**Specifications** Relevant Concept.

**NL Examples**

NL Template: "Think of what you need to prove and how you can connect that to the < Relevant Concept >."

- Occurrence state of definitions or substitutions: "Think of what you need to prove and how you can connect that to the powerset."
- Case distinction: "Think of what you need to prove and how you can connect that to the *union* $\cup$.", i.e., the disjunctively defined concept.
- Induction: "Think of what you need to prove and how you can connect that to *the set of all finite subsets of X*.", i.e., the inductively defined concept.
- Occurrence state of quantifiers: "Think of what you need to prove and how you can connect that to the *for all*.", i.e., the quantifier type.
- Occurrence state of connectives: "Think of what you need to prove and how you can connect that to the equivalence $\Leftrightarrow$.", i.e., the connective type.

  **Notes** In general, the Relevant Concept for case distinction is the disjunctively defined concept, for induction the inductively defined concept, for occurrence state of quantifier is the Quantifier type, and for occurrence state of connective the connective type.

  The hint is to be produced when the Relevant Concept is known.

`Give-away-subordinate-concept-meta-reasoning` It gives away the Subordinate Concept and explains its function, in finding the proof.

**Specifications** Relevant Concept, Subordinate Concept.

**NL Examples**

NL Template: "You can consider the < Subordinate Concept > and how it connects to the < Relevant Concept >."

- Occurrence state of definitions or substitutions: "You can consider the subset and how it connects to the powerset."
- Case distinction: "You can consider the different cases that you have to prove, deriving from the disjunctive definition of the *union* $\cup$.", where the < Relevant Concept > is the disjunctively defined concept, here the *union* $\cup$.

- Induction: "You can consider the different steps that you have to prove, deriving from the inductive definition of *the set of all finite subsets of X*.", where the < Relevant Concept > is the inductively defined concept, here *the set of all finite subsets of X*.

- Occurrence state of quantifiers:
  NL Template: "In order for the expression to hold for < Relevant Concept > we need to prove it for < Subordinate Concept >, as it is not possible to prove it for all x,y,..."

  "In order for the expression to hold for *for all* we need to prove it for some constants x,y,..., as it is not possible to prove it for all x,y,...", where the Relevant Concept is the Quantifier type and the Subordinate Concept the new goal formula.

**Notes** The Relevant and Subordinate Concepts are the Quantifier type and the new goal formula (i.e., the variables to be used), respectively.

The new hypothesis and the new goal formula and name are included in the Rule of Inference field for the realisation of the hint in the case of occurrence state of quantifier and connective.

- Occurrence state of connectives:
  NL Template: "In order for the expression with the < Relevant Concept > to hold, you have to prove < Subordinate Concept >."

  "In order for the expression with the equivalence $\Leftrightarrow$ to hold, you have to prove both directions of the expression", where the Relevant Concept is the connective type and the Subordinate Concept the new goal formula.

#### 4.5.2.2.3 Inference-Rule Meta-Reasoning Hints

`Elicit-domain-technique` It elicits the Domain Technique, as defined in Chapter 3.

**Specifications** Relevant Concept.

**NL Examples**

- Occurrence state of definitions or substitutions: "What should you do here in order to deal with the < Relevant Concept > (e.g., the powerset, or *if-then* relation)."

- Case distinction: "What should you do here in order to deal with the disjunctive definition of the < Relevant Concept >?"

- Induction: "What should you do here in order to deal with the inductive definition of < Relevant Concept >?"

- Occurrence state of quantifiers: “What should you do here in order to deal with the < Relevant Concept >” (i.e., the quantifier type).
- Occurrence state of connectives: “What should you do here in order to deal with the < Relevant Concept >?” (i.e., the connective type).

`Give-away-domain-technique` It gives away the Domain Technique, as defined in Chapter 3.

**Specifications** Domain Technique, Relevant Concept.

**NL Examples**

NL Template: “You have to < Domain Technique > the < Relevant Concept >.”

- Occurrence state of definition: “You have to get rid of the powerset.”, where the < Domain Technique > is extract.
- Case distinction: “You have to apply case distinction to the *union* $\cup$”, where the Relevant Concept is the disjunctively defined concept *union*.
- Induction: “You have to apply induction of *the set of all finite subsets of X*”, where the < Relevant Concept > is the inductively defined concept *the set of all finite subsets of X*.
- Occurrence state of quantifier: “You have to get rid of the *for all*.”, where the < Domain Technique > is extract and < Relevant Concept > is the quantifier type.
- Occurrence state of connective: “You have to get rid of the equivalence $\Leftrightarrow$.”, where < Domain Technique > is extract and the < Relevant Concept > is the connective type.

`Give-away-inverse-rule` It informs the student that the rule used is an inversion of the expected rule.

**Specifications** Inversion.

**NL Examples** “The rule that you want to apply is the < Inversion > inverse of the rule needed here.”

**Notes** Inverse rule, like basic knowledge is not an instructional point, but deals with domain mistakes related to the meta-reasoning of the Rule of Inference class only. It is defined in Appendix A. The corresponding hint is given when the student gives a rule and the relation inversion from our ontology holds. The NL realisation of the relation inversion might be different. Note that the active equivalent is not used, in the same way that other active hints relating to domain relations are not used.

This hint is motivated by our wizard-tutor’s comment (soc23k, T3), which expressed the wish to inform the student that she was starting from the wrong direction of the implication, instead of asking if she knows how to break the implication, which the student obviously didn't know.

Using the Inverse Rule by affirming the consequent is, in general, a very common mistake that humans make and its correction significantly improves problem-solving performance [Price *et al.*, 1997].

`Elicit-connect-relevant-subordinate-concept` It is given when the Domain Technique is not known and elicits the way the Relevant Concept is connected to the Subordinate Concept, by pointing to their function of this connection with regard to finding the Rule of Inference.

**Specifications** Relevant Concept, Subordinate Concept.

**NL Examples**

NL Template: "Think of a theorem or lemma that you can apply and involves the < Relevant Concept > and the < Subordinate Concept >."

- Occurrence state of definitions or substitutions: "Think of a theorem or lemma that you can apply and involves the powerset and the subset."
- Case distinction: "Think of a rule that you can apply and involves the disjunctively defined concept *union* $\cup$ and what you have to prove."
- Induction: "Think of a rule that you can apply and involves the inductively defined concept *the set of all finite subsets of* $X$ and what you have to prove."
- Occurrence state of quantifiers: "Think of a rule that you can apply and involves the *for all* and what you need to prove.", where the < Relevant Concept > is the quantifier type and < Subordinate Concept > is the new goal formula.
- Occurrence state of connectives: "Think of a rule that you can apply and involves the equivalence $\Leftrightarrow$ and what you need to prove.", where the < Relevant Concept > is the connective type and < Subordinate Concept > is the new goal formula.

**Notes** For occurrence state of quantifier the Relevant Concept and the Subordinate Concept are the quantifier type and the variables to be used, respectively. For occurrence state of connective the Relevant Concept and the Subordinate Concept are the connective type and what has to be proven, respectively.

`Give-away-connect-relevant-subordinate-concept` It gives away the Rule of Inference and explains why it applies, in connection to the Relevant and the Subordinate Concepts.

**Specifications** Relevant Concept, Subordinate Concept, Rule of Inference.

**NL Examples**

NL Template: "What connects the $<$ Relevant Concept $>$ and the $<$ Subordinate Concept $>$ is $<$ Rule of Inference $>$."

- Occurrence state of definitions or substitutions: "What connects the powerset and the subset is the definition of powerset."
- Case distinction: "What connects the disjunctively defined concept *union* and what you have to prove are the cases deriving from the definition of the *union*."
- Induction: "What connects the inductively defined concept *the set of all finite subsets of X* and what you have to prove are the inductive steps deriving from its definition."
- Occurrence state of quantifier: "What connects the *for all* and what you have to prove is the elimination of the *for all*.
- Occurrence state of connective: "What connects the equivalence $\Leftrightarrow$ and what you have to prove is the the elimination of the equivalence."

**Notes** The instantiation of the subordinate concept can be derived from the new goal formula.

`Elicit-elaborate-domain-object` It elicits the Rule of Inference by explaining its connection to the Domain Technique and the Relevant Concept.

**Specifications** Domain Technique, Relevant Concept.

**NL Examples**

NL Template: "Think of a theorem or lemma (a rule) that explains how to $<$ Domain Technique $>$ the $<$ Relevant Concept $>$."

- Occurrence state of definitions or substitutions: "Think of a theorem or lemma (a rule) that explains how to get rid of the powerset."
- Case distinction: "Think of a rule that explains how to apply case distinction to the disjunctively defined concept *union* $\cup$."
- Induction: "Think of a rule that explains how to apply induction to the inductively defined concept *the set of all finite subsets of X*."
- Occurrence state of quantifiers: "Think of a rule that would help you get rid of the *for all*."
- Occurrence state of connectives: "Think of a rule that tells you how to get rid of the equivalence $\Leftrightarrow$."

**Notes** It is given when the Domain Technique is known.

`Elicit-connect-relevant-subordinate-concept` gives away the Domain Technique together with the Rule of Inference, whereas in `elaborate-domain-object` the Domain Technique is already known and just repeated.

This hint was inspired by our wizard-tutor's comment that the Relevant Concept should be elaborated, as it is on its own not very helpful.

`Give-away-elaborate-domain-object` It gives away the Rule of Inference and explains why it applies in connection to the Relevant Concept and the Domain Technique.

**Specifications** Domain Technique, Relevant Concept, Rule of Inference.

**NL Examples**

NL Template: "What helps you < Domain Technique > the < Relevant Concept > is < Rule of Inference >."

- Occurrence state of definitions or substitutions: "What helps you get rid of the powerset is the definition of powerset."
- Case distinction: "What helps you apply case distinction to the disjunctively defined *union* $\cup$ are the cases deriving from the definition of the *union*".
- Induction: "What helps you apply induction to the inductively defined *set of all finite subsets of X* are the inductive steps deriving from its definition."
- Occurrence state of quantifiers: "What helps you get rid of the *for all* is elimination of the *for all*."
- Occurrence state of connectives: "What helps you get rid of the equivalence $\Leftrightarrow$ is elimination of the equivalence."

**4.5.2.2.4 Substitution Meta-Reasoning Hints** Meta-reasoning Substitution hints explain why we should substitute the particular terms given the domain information of the proof step and given the way we do substitutions, which is explained in the performable-step hint `give-away-substitution`.

`Elicit-inference-rule-application` It elicits the application of the Rule of Inference by requesting the substitution of the Relevant Concept.

**Specifications** Domain Technique, Relevant Concept.

**NL Examples**

NL Template: "What do you have to write down instead of the < Relevant Concept > to < Domain Technique > it?"

- Occurrence state of definitions or substitutions: “What do you have to write down instead of the powerset to get rid of it?”
- Case distinction: “What do you have to write down instead of the disjunctively defined *union* $\cup$ to apply case distinction to it?”
- Induction: “What do you have to write down instead of the inductively defined *set of all finite subsets of* $X$ to apply induction to it?”
- Occurrence state of quantifiers: “What do you have to write down instead of the *for all* to get rid of it?”
- Occurrence state of connectives: “What do you have to write down instead of the equivalence $\Leftrightarrow$ to get rid of it?”

`Give-away-inference-rule-application` It gives away the way the substitution has to preceed in connection to the Relevant Concept, the Domain Technique, and the Rule of Inference.

**Specifications** Domain Technique, Relevant Concept, Rule of Inference.

**NL Examples**

NL Template: “You have to $<$ Domain Technique $>$ the $<$ Relevant Concept $>$ by writing down $<$ Rule of Inference $>$ instead.’

- Occurrence state of definitions or substitutions: “You have to get rid of the powerset by writing down the definition of powerset instead.”
- Case distinction: “You have to apply case distinction for the *union* $\cup$ by writing down the cases deriving from the disjunctive definition of the *union* instead.’
- Induction: “You have to apply induction for *the set of all finite subsets of* $X$ by writing down the inductive steps deriving from its definition instead.”
- Occurrence state of quantifiers: “You have to get rid of the *for all* by writing down the rule for the elimination of the *for all* instead. ”
- Occurrence state of connectives: “You have to get rid of the equivalence $\Leftrightarrow$ by writing down the rule for the elimination of the equivalence instead.”

**Notes** Examples of instances of such hints in our corpus are:

- “You have to assume that the assumption holds and derive the proposition from this” (soc17k, T4)
- “You have to break down the relation by assuming the validity of the assumption to be able to prove the validity of the proposition” (soc21k, T4)

**4.5.2.2.5 Proof-Step Meta-Reasoning Hints** The Proof-step meta-reasoning hints address the step as a whole and not a subpart of it. Because of their overview nature, the production of some of these hints makes sense at the beginning of the hinting session to motivate the whole proof. Hence, these hints capture a cycle defined in the actual hinting algorithm. That is, the hinting session for a step starts with a proof-step meta-reasoning hint and finishes with a proof-step performable-step hint, namely `give-away-performable-step`.

`Elicit-starting-point` It elicits the reasoning for how one should tackle a problem.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "How can you start handling the problem?"

**Notes** This is a better realisation than the very general "Do you understand the problem". It avoids the abstractness that has been criticised in the case of Polya's instructions. "Do you understand the problem" is really a *check-origin-problem* in our dialogue-move taxonomy.

`Give-away-starting-point` It gives away the reasoning for how one should tackle a problem.

**Specifications** Starting Point.

**NL Examples** "You have to identify what is given and what you have to prove."

**Notes** The `starting point` passive and active hints are only produced at the beginning of the proof before the first step. The particular hints belong to meta-reasoning, as they address aspects that are not part of the proof, although they still comprise information used for deriving the proof step. It is characteristic that they are defined independent of any theory.

`Elicit-premise-conclusion` It elicits the premise/-s of the problem at hand and the desired conclusion.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "What is assumed here and what do you have to prove?"

`Give-away-premise-conclusion` It gives away the premise/-s and the conclusion.

**Specifications** Premise-Conclusion.

**NL Examples** "What is assumed is $<$ Premise $>$ and what you have to prove is $<$ Conclusion $>$."

**Notes** Note that, in case of a connective, once the connective has been eliminated, the expression to be proved becomes one that either does not include a connective, or the connective is different. In the latter case, the source and the target for the following step are defined accordingly. Only if the next step is a subproof would these coincide with the premise and the conclusion of the proof.

`Elicit-abstract-method` It elicits the Abstract Method, that is whether a direct or an indirect proof applies.

**Specifications** Target.

**NL Examples** "You have to decide now if you should try to prove the $<$ Target $>$ or assume the opposite of it?"

**Notes** This hint is produced when the proof is indirect.

Both `elicit` and `give-away-abstract-method`, which follows, address the instructional point proof method in our domain. They capture the method that is not particular to the specific theory in the domain, but methods that are more generally used in the problem solving of the domain.

`Give-away-abstract-method` It gives away the Abstract Method.

**Specifications** Abstract Method.

**NL Examples**

- Direct proof: "You have to try to manipulate what is given in order to reach your goal."

  **Notes** This case is included only for the case that the student asks about it with a *request-assistance*.

- Indirect proof: "You have to assume the validity of the opposite of the $<$ Target $>$ and prove a contradiction based on what you know."

`Elicit-specific-method` It elicits the Specific Method for tackling the problem. In our domain whether a forward or a backward step applies.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "Ask yourself in which ways you can manipulate an expression to prove what you want and choose the right one to apply here. "

`Give-away-specific-method` It gives away the Specific Method for tackling the problem.

**Specifications** Specific Method.

**NL Examples**

- Forward steps: "You manipulate an expression by using what you know and applying rules that help you get to what you want to prove"
- Backward steps: "You manipulate an expression by simplifying what you want to prove"

`Give-away-step-meta-reasoning` It gives away a summary of the meta-reasoning for the whole step by explaining the main instructional points for deriving the step.

**Specifications** The passive equivalent of all meta-reasoning hints produced for the step already, plus the central ones from the rest. These are:

- `give-away-premise-conclusion`
- `give-away-specific-method`
- `give-away-relevant-concept-meta-reasoning`
- `give-away-domain-technique`
- `give-away-elaborate-domain-object`
- `give-away-inference-rule-application`

**NL Examples** "The reasoning for this step is as follows: First, we have to simplify the expression. We want to show that *if* $A \subseteq K(B)$*, then* $B \subseteq K(A)$ holds, by proving $B \subseteq K(A)$ under the assumption that $A \subseteq K(B)$. We concentrate on the *if-then* relation, which can help us simplify the expression to prove the conclusion. We try to think of a rule that will help us eliminate the *if-then*, so we apply the rule if X $\Rightarrow$ Y, then let X and prove Y".

**Notes** This is a recapitulation of the meta-reasoning for a whole proof step. It is an instructional point as it addresses the explanation of the proof step as a unit. Its active form is the same as an *align*, so it is not used.

The hint was motivated by the complaint of several subjects in our experiment that they often lost the overview of the proof.

#### 4.5.2.3 Pragmatic Hints

Pragmatic hints always give away the respective pragmatic information. However, in the domain knowledge dimension they address whatever domain information is being dealt with at the point they are produced. This way, pragmatic hints also implicitly follow the passive-active distinction, as far as the hint category gives some information away in order to elicit some other piece of information. By the same token, pragmatic hints also aim at eliciting performable step or meta-reasoning information. The classes for pragmatic hints indicate the pragmatic cognitive function of the hints. The choice of the hint category by the algorithm reflects the choice of the class, as well. We now look into the different classes of pragmatic hints and the hint categories they include.

**4.5.2.3.1 Speak-To-Answer Hints** Speak-to-answer hints refer to the preceding answer of the student and comment on pragmatic aspects in it. They pick at a particular point about it that gives rise to the specific hint.

`Ordered-list` It informs the student that there is an element missing from the expected answer, which is a list. It specifically refers to the order in the list in which the expected answer appears.

**Specifications** List Position.

**NL Examples** "You are missing the $<$ List Position $>$ (e.g., the third case)", where the list is the cases of the case distinction.

Or "You still have to figure out a $<$ List Position $>$ (e.g., a third point)."
.

**Notes** Examples of lists are the cases of the case distinction and the steps of the induction. The element missing is one or more of the cases or steps respectively.

`Unordered-list` It points the student to the fact that the expected answer corresponds to a list and prompts for its missing elements in the student's answer. It only refers to the number of elements in the list.

**Specifications** List Elements, Rest List Element.

**NL Examples**

- "You have now $<$ Rest List Element $>$ (e.g., two cases), but you need $<$ List Elements $>$ (e.g., three). Which one are you missing?"
- "And what more?"

**Notes** The difference to `ordered-list` is that the list has no particular order, and so the missing element/-s are not referred to with respect to any order either.

`Narrow-down-choice` It helps the student find the expected answer by restricting the search space. Examples of `narrow-down-choice` can be abstract-method and specific-method, where there is only a limited number of methods available.

**Specifications** List Elements.

**NL Examples** "You can do < List Elements > (e.g., case distinction, or induction). Which one applies here?"

**Notes** This hint is connected with what [Wood and Middleton, 1975] call "reduction of degrees of freedom" (p. 98). They describe the way tutors reduce the possible moves, in order to allow the student the possibility to complete it.

`Discrepancy` It refers to a discrepancy between the student's answer and the expected answer.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "Really?"

**Notes** This simple NL realisation was used by our wizard-tutor. [Wood and Middleton, 1975] call this kind of feedback "marking of critical features" (p. 98).

**4.5.2.3.2 Point-to-Information Hints** These hints point the student to some information given previously, either during the dialogue or in the lesson material. They make the relation between the hint and the information it addresses with its context more obvious. They also make the student more aware of the material at their disposal, leaving them nonetheless free to choose the exact way they will use it. There is some element of metacognition here, but we are not on the whole addressing metacognition.

`Point-backwards` It refers the student to a previous use of the information at hand to reactivate it and highlight its context.

**Specifications** Same Domain Information, that is the information that the tutor points back to.

**NL Examples** "We have already looked at a similar case in this proof. Have a look back and try to move on with the proof."

**Notes** This hint was also produced by our wizard-tutor.

It brings to memory the original context where the relevant information was discussed [Price *et al.*, 1997], and it enables assimilation of new information in existing schemata [Widmayer, URL].

Currently in `Menon`, we implement giving away the information that is the same if the student can still not find it after the `point-backwards`. Same Domain Information comprises the content of any of the passive conceptual hints (cf. Chapter 6).

`Refer-to-lesson` It points the student to some specific piece of information in the lesson material, which comprises the current expected answer.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "Look it up in your lesson material!"

**Notes** The pedagogical motivation of this pragmatic aspect is that the student is pointed to consulting the available material better, while being at the same time directed to the piece of information currently needed for the task. This information can be anything available in the lesson material; from a concept, to the Rule of Inference, to an example, etc. The aim is to make the connection to the current situation obvious.

`Point-to-lesson` It points students to the lesson material as a whole and asks them to read it again.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "Please, read your lesson material carefully once more. Let me know when you are ready."

**Notes** This hint is produced when it appears that students cannot be helped by tutoring, probably because they have not read the material carefully.

**4.5.2.3.3 Take-For-Granted Hints** Take-for-granted hints ask students to accept some information without further explanation or elaboration, for example, because that would require to delve into another mathematical topic, which would shift the focus of the tutorial session. This is motivated by the notion of "local axiomatics" [Wu, 2001], prominent in teaching mathematics. It was also recognised by [Wood and Middleton, 1975] who included it under the tutorial goal with the name "direction maintenance" (p. 98).

`Correct-information` It asks the student to accept a chunk of information.

**Specifications** Only NL specifications are applicable based on the hint category.

**NL Examples** "Let's not look into this now."

`Correct-term` It provides the correct terminology to be used, which is not known by the student.

**Specifications** Correct Linguistic Term, Wrong Linguistic Term.

**NL Examples** "Just say < Correct Linguistic Term > instead of < Wrong Linguistic Term > to be accurate."

**Notes** The correct linguistic term can be "element of", correcting a wrong linguistic term "belongs to", which was used in our corpus when the hint was produced by our wizard-tutor.

`Misconception`

**Specifications** Misconception Type.

**NL Examples** "Keep in mind that...(explanation of misconception follows, based on the < Misconception Type >.)"

**Notes** We allow the possibility of treating misconceptions in our teaching model and we include its treatment in our Socratic teaching strategy. We have defined misconception exactly from the point of view of the teaching strategy. In Chapter 5 we clarify what counts as a misconception in `Menon`. The explanation of the misconception is dependent on the misconception type.

## 4.6 Subdialogues and Subtasks

We now define subdialogue and subtask as the units where dialogue moves reside.

A subdialogue or the subtask is initiated by a dialogue move when a subdialogue or a subtask is the result of the move having been performed. For instance, a *request-clarification* initiates the clarifications subdialogue that follows it and a *request-assistance* can initiate a subtask to deal with the request. Note that conventional-management dialogue moves, such as *initiate-subdialogue*, do not themselves initiate a subdialogue or a subtask, but signal its initiation.

**Subdialogue** A *subdialogue* is defined at the dialogue level. It comprises every side-tracking necessary to continue with the tutoring task. As a result subdialogues are initiated by either student or tutor turns. When the tutor cannot categorise the student input and performs a clarification move, a subdialogue is initiated. The student can initiate a subdialogue by, e.g., a *request-clarification*. *Check-origin-problem* initiates a subdialogue, no matter how it might be realised at the forward-looking level. The move that initiates it is neither a *domain-contribution* that can be categorised for its contribution to the task, nor any dialogue move that comprises domain feedback. For example, *align* does not

| **Proof and Proof Step Information** | **Meta-reasoning vs. Performable Step** | **Active vs. Passive** | **Pragmatic vs. Conceptual** |
|---|---|---|---|
| domain relation | domain relation | | |
| antithesis<br>duality<br>conversion<br>hypotaxis<br>specialisation<br>generalisation | antithesis<br>duality<br>conversion<br>hypotaxis<br>specialisation<br>generalisation | | |
| *domain object* | *domain object* | *elicit* | *speak-to-answer* |
| rel-con<br>sub-con | rel-con-meta-reas<br>sub-con-meta-reas | elicit-rel-con<br>elicit-start-point<br>. . .<br>. . . | unordered-list<br>ordered-list<br>narrow-down-choices<br>discrepancy |
| *rule of inference* | *rule of inference* | *give-away* | *point-to-information* |
| inference-rule<br>basic-knowledge | connect-rel-sub-con<br>elaborate-domain-object<br>inverse-rule<br>domain-technique | give-away-rel-con<br>give-away-start-point<br>. . .<br>. . . | point-backwards<br>refer-to-lesson<br>point-to-lesson |
| *substitution* | *substitution* | | *take-for-granted* |
| substitution<br>ill-formed | inf-rule-application | | correct-info<br>correct-term<br>misconception |
| *proof step* | *proof step* | | |
| | starting-point<br>abstract-method<br>specific-method<br>premise-conclusion<br>proof-step-meta-reas | | |

Table 4.1: Summary of taxonomy of hints with dimensions representation and examples of hint categories in them

1. **Domain Information - Performable Step - Conceptual - Active vs. Passive**
    - *Domain-object Hints*
        - elicit-relevant-concept
        - give-away-relevant-concept
        - elicit-subordinate-concept
        - give-away-subordinate-concept
    - *Rule-Of-Inference Hints*
        - elicit-inference-rule
        - give-away-inference-rule
        - elicit-basic-knowledge
        - give-away-basic-knowledge
    - *Substitution Hints*
        - elicit-substitution
        - give-away-substitution
        - elicit-ill-formed
        - give-away-ill-formed
    - *Proof-Step Hints*
        - elicit-proof-step
        - give-away-proof-step
2. **Domain Information - meta-reasoning - Conceptual - Active vs. Passive**
    - *Domain-Relation Meta-Reasoning Hints*
        - give-away-antithesis
        - give-away-duality
        - give-away-conversion
        - give-away-hypotaxis
        - give-away-specialisation
        - give-away-generalisation
        - give-away-primitive
    - *Domain-Object Meta-Reasoning Hints*
        - elicit-relevant-concept-meta-reasoning
        - give-away-relevant-concept-meta-reasoning
        - elicit-relevant-concept(subordinate-concept)
        - give-away-relevant-concept(subordinate-concept)
        - elicit-subordinate-concept-meta-reasoning
        - give-away-subordinate-concept-meta-reasoning
    - *Rule-Of-Inference Meta-Reasoning Hints*
        - elicit-domain-technique
        - give-away-domain-technique
        - give-away-inverse-rule
        - elicit-connect-relevant-subordinate-concept
        - give-away-connect-relevant-subordinate-concept
        - elicit-elaborate-domain-object
        - give-away-elaborate-domain-object
    - *Substitution Meta-Reasoning Hints*
        - elicit-inference-rule-application
        - give-away-inference-rule-application
    - *Proof-Step Meta-Reasoning Hints*
        - elicit-starting-point
        - give-away-starting-point
        - elicit-premise-conclusion
        - give-away-premise-conclusion
        - elicit-abstract-method
        - give-away-abstract-method
        - elicit-specific-method
        - give-away-specific-method
        - give-away-step-meta-reasoning
3. **Pragmatic**
    - *Speak-To-Answer Hints*
        - ordered-list
        - unordered-list
        - narrow-down-choices
        - discrepancy
    - *Point-To-Information*
        - point-backwards
        - refer-to-lesson
        - point-to-lesson
    - *Take-For-Granted*
        - correct-information
        - correct-term
        - misconception

Figure 4.3: Summary of the taxonomy of hints

initiate a subdialogue, because the task carries on as before. It constitutes the feedback, when the tutor can categorise the *domain-contribution* and chooses *align* only to make sure that the student really understands. Subdialogues initiated by the student are not treated by `Menon`.

**Subtask** A *subtask* is defined at the task level and contrasts with the main task, which is tutoring the proof. The move initiating it constitutes tutoring feedback. A subtask comprises a set of dialogue moves that aim at fulfilling the same subgoal at the proof-task level, whose goal is to help the student find the proof. The fact that the tutor employs a subtask means that the choice of tutoring feedback was possible, and thus the tutoring task is running smoothly. Any task dialogue move can initiate a subtask.

All moves can be part of subtasks or subdialogues. A subdialogue can take place within a subtask without influencing the flow of the task itself. The relation among subtasks is dependent on the teaching strategy.

**General Remarks on the Hint Taxonomy** In this section, we have only seen combinations of the four dimensions that are motivated by our teaching model. However, combinations of aspects like active, conceptual, domain-relation, and performable-step would serve the specific purpose of explicitly teaching such relations in the form of declarative knowledge, which is not one of our tutorial goals. Such hints would elicit the relation between two mathematical objects in the proof step (e.g., the converse between $\subseteq$ and $\supseteq$). The passive counterpart, in contrast, can be used to elicit, for example, the Relevant Concept. If the student mentioned $\subseteq$ instead of $\supseteq$ a hint could be formulated like: "Not really $\subseteq$, but something closely related."

## 4.7 Natural Language Automation of Hints

The specification of the tutor task-dialogue moves in Section 4.4.2 are provided for the NL realisation of hints. They may affect the exact realisation in conjunction with other dialogue model consideration every time. An example of how this may relate to the tutor model is given in the analysis of *encourage* above. This thesis provides a basis for such analysis, but does not undertake it.

Although we have provided examples of hints formulated in natural language, we do not make any claims about the sentence-level realisation of the hint categories. These examples were only produced to help the reader picture the final production of hints. We have implemented such templates for NL formulations of hints in `Menon` where the domain information can be instantiated by a domain reasoner. As usual in a cascaded NL generation process, this provides an interface and more or less any of the existing systems could be connected to `Menon` via this interface.

However the aim of this thesis is to provide the means for automatically producing hints. For this purpose, we also provide the full representation language of `Menon`'s input and output in Appendix B. The input fields are provided by either the domain reasoners (proof manager, math KB) or, in a few cases, by the sentence analyser, and can be passed on to the generator as parameters of the hint categories. They can be requested from the responsible domain reasoner after the hint algorithm has determined the hint category and its specifications. This way the domain reasoner only has to calculate the information necessary for the hint generation. The ontology can be used by the domain reasoners to instantiate the instructional points for the current step. The current step itself is found by comparing the proof to the student input. Note that the student sees only the natural language realisation for the instantiations of the specifications that are produced by the generator. The NL generator can then takes dialogue and discourse aspects into account and can provide a contextual natural language realisation of the produced hints.

## 4.8 Relation of Hints to other Dialogue Moves

This section provides some theoretical observations to illustrate the relation of hints as task-dialogue moves to the other move functions in the dialogue-move taxonomy.

To start with some general remarks, hints should be realised by the dialogue move that most precisely asks for the content of the hint in order not to confuse the student with imprecise questions. For instance, indirect formulations of the sort "Do you know ..." should be avoided. Students may take them literally and reply with "yes" or "no", failing to abide by the obligations of the genre.

*Hints* have more or less the same communicative function, so there probably is a direct relationship between the optimal realisation of *hints* and the hint function in a given dimension of the hint taxonomy. An example of this is the hint dimension of ELICITATION STATUS, which distinguishes between the functions active and passive. Active hints are either *asserts*, communicating a claim about the world [Allen and Core, 1997] or *diagnostic-queries*, testing whether students know a piece of information by asking them to supply the information (ibid.). Passive *hints* could be *asserts*, *open-options*, that is utterances that suggest a course of action or state a possibility (ibid.)[6], or *action-directives*, which are utterances that request an action to be performed. *Assert* means in this case that the tutor gives some piece of information away. *Open-option* (e.g., "Try to use P") and *action-directive* (e.g., "Use P") are used when the tutor offers a solution and probably the worse the student performance the more appropriate an *action-directive* would be.

Note that these possibilities span the different dimensions of dialogue-move functions. That is, hints of a specific cognitive function may also have a function in each of the dialogue-move dimensions, e.g., a *hint* in the task dimension, an *assert* in the forward-looking dimension, and so on. These functions all influence the final NL realisation of the hint.

[6]For as long as the student is not totally committed to a proof, *hints* are likely to be *open-options*.

We implement the idea of multiturn hints by subtasks like `spell-out-task`, which capture the fact that a trail of thought is carried through the multiple turns. However, the trail of thought itself does not constitute the basic unit of instruction, but our hints and task dialogue moves do. This should not be confused with multiutterance hints, which just means that it might be better to break a hint category into more than one turn for its NL realisation.

We will now consider some confusing cases, which seem to disobey the difference between hints and other task dialogue moves, as well as some cases of relations of dialogue moves between them. We include the decision trees for the complicated dimensions FORWARD LOOKING and BACKWARD LOOKING that are analysed in Appendix C for quick reference. The decision trees in Figures 4.4 and 4.5 capture the relation of the dialogue moves within the same dimension in that they represent how one can decide on which of the functions described in a dimension are fulfilled by an utterance[7].

**Info-request vs. Diagnostic-query** An *info-request* is performed when the speaker really does not know the answer to the content of the question involved. For example, in the *info-request* 4.43 (soc1p), the wizard-tutor did not know what the student was getting at. On the contrary, a *diagnostic-query* is a test for the student's knowledge when the tutor knows the relevant information. In the *diagnostic-query* in 4.44 (soc1k), the tutor knows that the powerset has nothing to do with this proof, but wants to check to what extent the student realises this. A subtask is initiated by this move.

(4.43) **T4:** . . . , aber wie bringt das den Beweis weiter?
*. . . , but how does this help with the proof?*

(4.44) **T2:** ". . . , aber was hat die Potenzmenge mit diesem Beweis zu tun?"
*. . . , but what does the powerset have to do with this proof?*

**Hint vs. Check-origin-problem** The difference between *check-origin-problem* and *hint* is that a *hint* at forward-looking level can be a *diagnostic-query*, whereas a *check-origin-problem* will most commonly be an *info-request* as the tutor often really does not know the answer to the question "What is wrong?", "Can you not go on?".

**Acknowledge vs. Signal-dom-con-evaluation** Another interesting case is *acknowledge*. 'Acknowledgements are utterances consisting of short phrases such as "ok", "yes", "uh-huh", that signal that the previous utterance was understood without necessarily signalling acceptance [Allen and Core, 1997]. It is important from a tutoring perspective, because of the obligations involved. Namely, when the student addresses the tutor, the tutor is obliged to *acknowledge* it, even if it is only to reject the offer to talk about the issue that the

[7]The decision tree is partly based on the ones provided in [Allen and Core, 1997] and [Core *et al.*, 2002].

student has raised. The preferred way of doing the latter is in an explicit manner that would be too condescending and evasive in other genres: "Before we get to that,..." An explanation for that is that, because the tutor does not give direct answers, an indication is on call to show that the student's answer is taken into account and is not just ignored.

However the student is not obliged to do any explicit *acknowledge*. Sometimes they don't do any implicit *acknowledge* either. They just go silent and assume that the tutor knows that they are listening and thinking about the problem. The assumption is that in any other case the student would ask for clarification. An explanation for that is that both students and tutor are aware of the obligations that their respective social roles bring. Since it is the students' obligation to take what the tutor, as the expert, says into account students do not feel that they need to indicate that they are doing so.

**Align vs. Discrepancy vs. Check-origin-problem** With *align*, the person who knows more tries to get evidence that the interlocutor shares the same knowledge, as in Example 4.45 (soc21k).

(4.45) **T6:** "...Warum haben wir zuerst die wenn-dann Beziehung betrachtet?"
*[Why did we deal first with the* if-then *relation?].*

`Discrepancy` is a pragmatic-hint. There are two possible ways of realising it. The first is a simple "Why?" or "Really?", that attempts to make students realise that they have made a mistake, without pointing out the specific mistake, as in Example 4.46, **T6** (Soc20p). Following this hint, the student indeed corrects herself in **S6**.

(4.46) **S5:** "entschuldigung , es gilt natürlich: $P ( C \cup ( A \cap B ) ) \subseteq P ( C ) \cup P ( A \cap B )$< /s >"
*[I'm sorry, it holds of course: $P ( C \cup ( A \cap B ) ) \subseteq P ( C ) \cup P ( A \cap B )$< /s >]*
**T6:** "Wirklich?" *[Really?]*
**S6:** "nein doch nicht...andersrum."
*[no indeed not...the other way round.]*

This is appropriate when the mistake is one symbol, but so central, that it makes the whole *domain-contribution* wrong.

In this case, saying "Do you really mean that", could be mistaken for a *check-origin-problem*. This realisation disambiguates it, showing that the tutor is doubting the correctness of what the student wrote.

The other possible realisation of `discrepancy` points to the part of the answer that is wrong in order to allow the student to correct it (self-correction). e.g., "Did you really mean x?", "Can you clarify what you mean by x?". `Discrepancy` can be realised as a *request-clarification* at the backward-looking level, and as a *diagnostic-query* at the forward-looking level. Other possibilities of realising `discrepancy` at the forward-looking level are, as with all hints, the following:

- By an *action-directive*, e.g., "Check your answer again."
- By an *open-option*, e.g., "Try to correct your answer."

Note that the cognitive function of the two realisations is the same. They try to find out if the student really made a conceptual mistake or not, but still point out that there was a mistake. Finally, `discrepancy` does not start a subdialogue or a subtask, as it constitutes feedback that is based on the evaluated domain contribution (cf. Section 4.6).

*Check-origin-problem*, on the other hand, (e.g., "Can you clarify what you mean?" "What do you mean?") does not give away any information and, hence, it is not a *hint*. When performing it, the tutor is certain that there is a conceptual mistake, but is not exactly sure what it is and, therefore, cannot categorise the *domain-contribution* in order to give the appropriate feedback. *Check-origin-problem* realised as a *signal-non-understanding* in the backward-looking dimension (e.g., "Can you check your answer again?") expresses exactly this inability to understand. Otherwise, it may be realised by a *request-clarification*, as in Example 4.47 **T3** is from the BE&E corpus, and Example 4.48 **T3** from our corpus.

(4.47) **T1:** "OK, now what is the miliampmeter telling you about? Current, voltage, or resistance?"
**S2:** "resistance"
**T3:** "what is it that makes you think so?"
**S4:** "the directions were talking about resistance and miliamps"
**T5:** "Look again at the subsubsection that talks about playing with difference values on the rheostat..."

(4.48) **S2:** "$A \not\subseteq B$"
**T3:** Warum? *[Why?]* (soc13k)

In Example 4.48 **S2**, although the student's *domain-contribution* was not incorrect or irrelevant, as it was not what was expected at that point in the proof, but it might appear as a valid later step. Therefore, the tutor could not categorise it and asked "why".

**Hint vs. Domain-contribution** A hint attempts to resolve the proof task or issues related to the proof task indirectly and characteristically not to the best of the speaker's ability. A *domain-contribution*, on the contrary, tries to advance the proving task directly and to the best of the speaker's ability.

To illustrate that the same hint category could have different realisations, let's examine an example of a hint. Example 4.50 shows a *hint* in the task dimension and it has a passive relevant-concept hint function. The particular realisation is an *open-option* in the forward-looking dimension. Given a different dialogue context, however, the same underlying hint function could just as well be an *action-directive* without altering the hint function at the cognitive level, but serving the dialogue context better, as in the constructed Example 4.50. This approach is contrary to previous attempts to model tutorial dialogues, which do not distinguish between the cognitive and dialogue function of hints and have, thus, provide no clear account of the status of hints in the proposed dialogue-move taxonomies [Person and Graesser, 2003; Core *et al.*, 2002].

(4.49) **T2:** "Sie müssen als erstes die wenn-dann-Beziehung betrachten." (soc17k)
*[First you have to consider the* if-then *relation.]*

(4.50) **T:** "Consider first the *if-then* relation." (constructed)

## 4.9 Conclusion

We have presented a dialogue-move taxonomy, which attempts to clearly separate the general dialogue-management attributes of utterances from the genre and domain specific ones. We looked at how such a separation facilitates a deeper understanding of the appropriate manipulation of the genre specific phenomena and showed how it allows their formalisation. That in turn, enables building a reusable and easily reconfigurable dialogue manager, based on such principles.

In terms of the relationships between the task dimension and moves in other dimensions, the separation suggested here allows us to capture the special status that moves have for the tutorial-dialogue genre. As task-dimension moves are commonly indirect speech acts that attain their function based on the conventions in the genre, it is highly unlikely that they serve the same purpose in other genres. For instance, a *signal-non-understanding* at the backward-looking function can be read as the student asking for assistance, that is, as a *request-assistance* at the task dimension. It is necessary to represent and recognise the move, albeit not to commit as to the tutoring feedback that will be produced to treat it. Representing such indirect dialogue moves, in general, allows their easy manipulation according to the desired teaching model, without restricting the choice of the model. The task dimension of the dialogue-move taxonomy captures this flexibility. In Chapter 5 we explain how the Hinting Session Status represents the information that is relevant to the tutoring task. The specific context for the recognition and planning of the perlocutionary force of dialogue moves in task-oriented dialogues is the task itself [Traum, 1999]. Therefore, the Hinting Session Status is input to the Socratic procedure, which uses the tutoring-task information represented in it to determine the task-dialogue moves to be produced as tutorial feedback (cf. Chapter 6).

Especially within the task dimension, the analysis of the functions of hints facilitates the main goal of this thesis. Namely, to capture the cognitive function of hints and provide the necessary hint specifications for the generator, but allow the freedom to the dialogue manager and the NL modules to manipulate the phrasing. This opens up the path to integrating NL capabilities into an ITS as in the LeActiveMath project [Callaway *et al.*, 2006] that we reviewed in Chapter 1, but also to additionally preserve the benefits of sophisticated tutorial feedback.

The analysis of hints that we presented is an attempt to advance the understanding of hints and attempts to automate hinting. On the whole, the ELICITATION STATUS dimension is the only one that most other approaches capture explicitly through designing sets of related hints with increasing degrees of precision in revealing the required information. The three dimensions DOMAIN KNOWLEDGE, INFERENTIAL ROLE, and PROBLEM REFERENTIAL PERSPECTIVE are typically merged grossly under what we separate out as the ELICITATION STATUS.

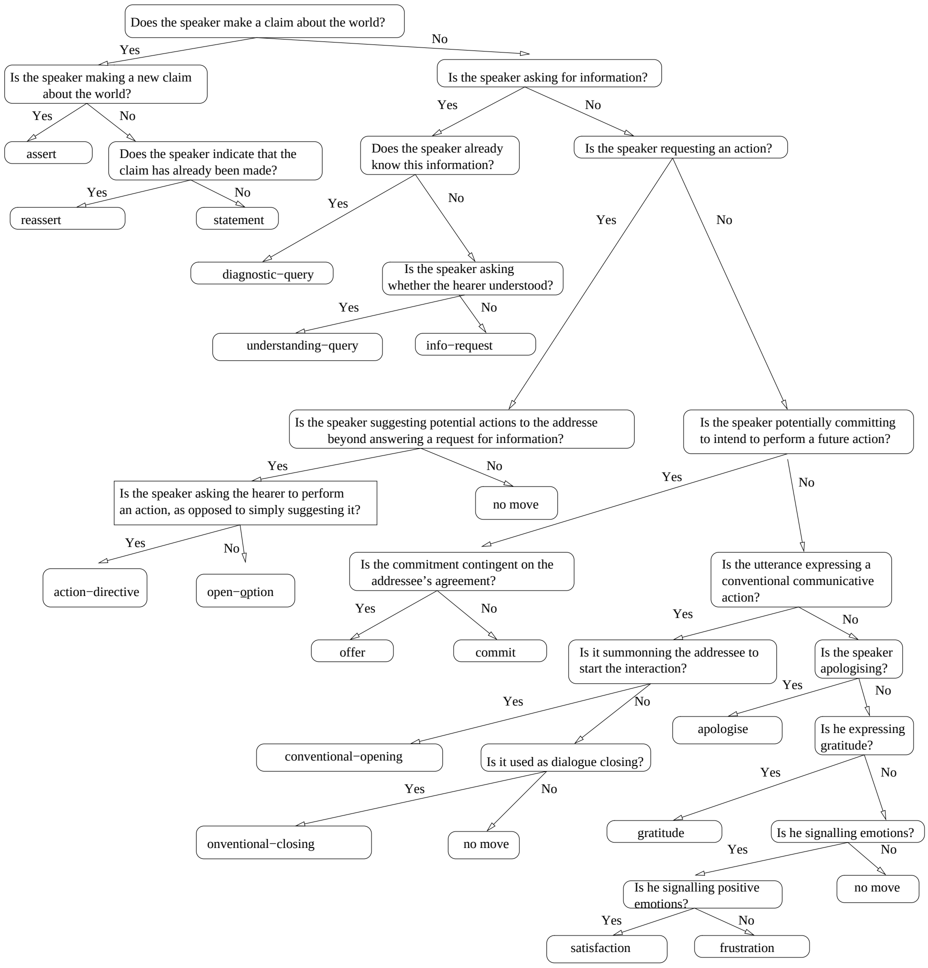


Figure 4.4: Forward-looking function: decision tree

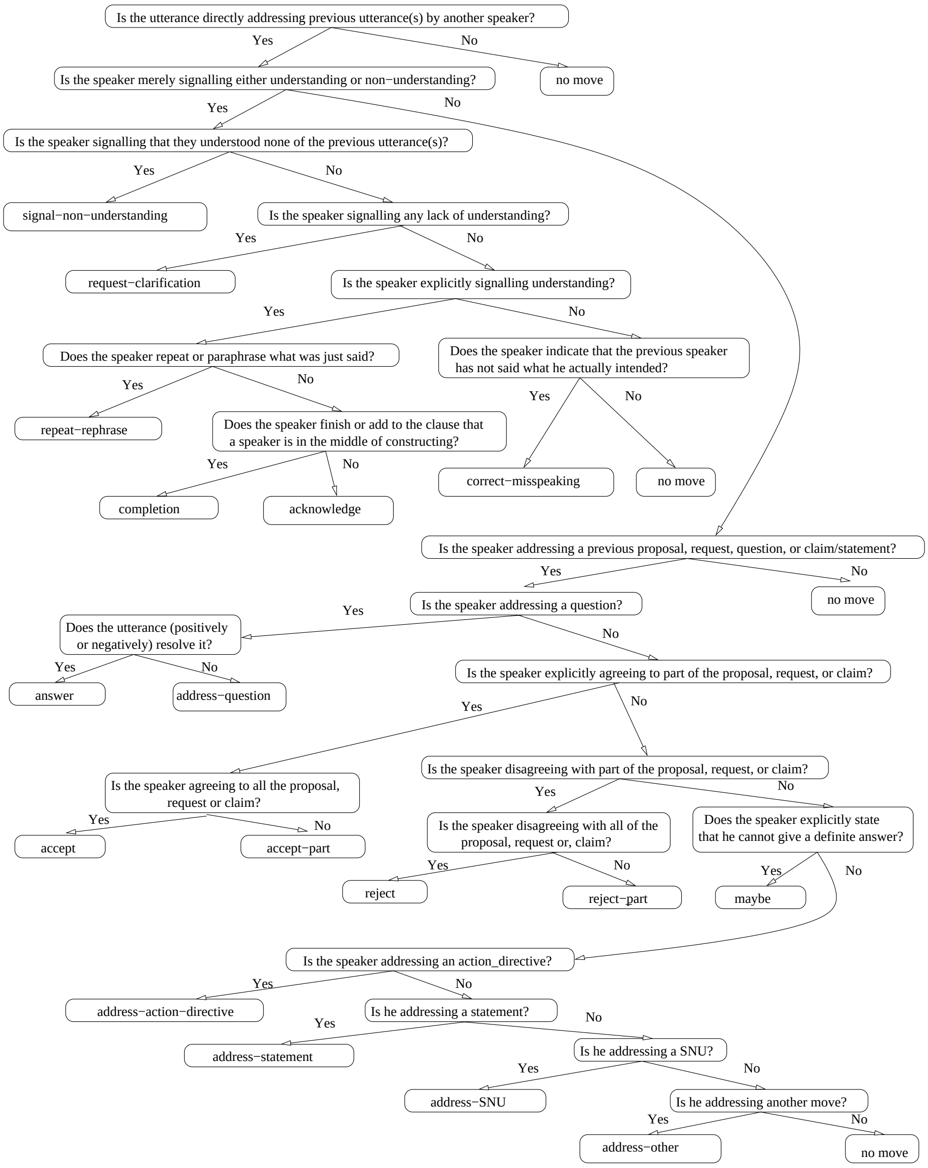


Figure 4.5: BACKWARD-LOOKING function: decision tree

# Chapter 5

# Hinting Session Status

## 5.1 Introduction

In order to provide adaptive feedback, we need to have a model of the progress of the student in the hinting session. In other words, we need to take into account various aspects of the student's performance, interpret them, and mould the feedback in accordance. We define the *hinting session* as a session where one proof task is being dealt with, from the moment the tutor sets the task, until it is completed.

The student modelling necessary in a tutoring system can then be divided into two types:

1. The model commonly referred to as the *user model*. It corresponds to what VanLehn [VanLehn, 2006] calls the outer loop, that is, the choice of problem to be assigned as a task for the student. Therefore, it is static during one hinting session. It informs such choices as the kind and level of the proof task, the level of abstraction allowed, the domain knowledge that the student should already possess, etc[1].

2. A model of the student while performing one task, which VanLehn refers to as the inner loop. It influences the choice of feedback to be provided for each problem step. This model is dynamically updated within one hinting session. It represents knowledge on the student's performance for the session and is not relevant to other sessions.

Since the aim of this thesis is to automate the intra-session feedback, this chapter concerns itself with the second type of student model. In `Menon` this is called *Hinting Session Status*.

[1] These choices are part of our global tutorial goal (See Chapter 2.)

## 5.2 Motivation of HSS

The *Hinting Session Status* (HSS) is a collection of parameters relevant to tutoring decisions during one hinting session. The set of values of the parameters at every point defines a *tutoring situation*.

The HSS can be thought of as the pedagogical representation of the proof task. It is based on the proof representation used in the task manager of the dialogue manager (cf. Chapter 1). It captures information that is of pedagogical value, because of the inferences that can be based on it. In a nutshell, this is information about the development of the proof and tutoring task, which allows making assumptions about the student's level of understanding, motivation and cognitive load. Therefore, feedback can be based on it.

The parameters of the HSS derive from our tutorial goals and the teaching model that we defined to achieve those goals (cf. Chapter 2). Consequently, we represent those aspects of the student's performance that help us recognise a certain behaviour. This is the behaviour that according to our teaching model requires treatment. Treatment takes the form of feedback, and aims at remediating problems, encouraging the existing desired behaviour, or promoting new desired behaviour, all towards the goal of promoting good proving and learning practice.

The organisation of the HSS reflects our tutorial goals and comprises the following classes with their subclasses, which we analyse in the next section:

1. External Tutoring Management
2. Tutoring History
3. Tutorial Goal Status
   (a) Instructional Points
   (b) Pragmatic Information
   (c) Proof Status
4. Motivation and Cognitive Load
   (a) Global Motivation and Cognitive Load
   (b) Local Motivation and Cognitive Load

The Socratic procedure, examined in Chapter 6, takes the HSS as input and is then responsible for interpreting the hinting situation and deciding on the appropriate feedback.

## 5.3 Hinting Session Status Fields

The HSS fields (see Table 5.1 for quick reference) represent the parameters that influence the pedagogical behaviour of the tutorial manager. Some of the parameters comprise several categories with subgategories. They are used to

| **HSS fields** | | |
|---|---|---|
| **Class** | **Subclass (if applicable)** | **Fields** |
| **ETM** | – | task, strategy, undo |
| **TH** | – | previous hint, substrategy, previous substrategy, tutor task dm |
| **TGS** | – | student-task dm |
| | – | previous student task dm |
| | – | domain-contribution |
| | Instructional points | relevant concept used, subordinate concept used, domain relation used, rule of inference used, technique used, starting point used, premise used, conclusion used, directness used, direction used |
| | Pragmatic points | different theory, same domain information, ordered list, unordered list |
| | Proof status | proof step complete, proof complete |
| **MCL** | GMCL | proof-step-C, unknown-C, misconception-C, missing-basic-knowledge-C, request-asssistance-C, resign-C, wrong-C, step-size-C, irrelevant-C, ill-formed-C, wrong-linguistic-term-C, time-out-C, domain-contribution-C |
| | LMCL | local-wrong-C, local-correct-C, local-domain-contribution-C, local-hint-C |

Table 5.1: Table of the HSS fields without their instances.

model the student's performance in a tutoring session. Not all dialogue moves at the TUTORING-TASK level have a bearing on the task at hand (i.e. the proof) the way hints do. They might, however, require general pedagogical feedback, which can be realised via another dialogue move. This influences learning in general, but is not specific to proving. For example, a request by the student to evaluate her performance, which we call *request-evaluation*[2], does not affect the hint choice, but it is taken into account by the Socratic strategy for the kind of feedback to produce. It is, therefore, represented in HSS. Prompting the student is another example of such pedagogical feedback. *Prompts* often accompany a hint. The student is asked to take the hint into account and try to make use of it in order to proceed with the task.

We are now going to inspect the fields of the HSS. For every one of the fields we include:

1. Explanations of what the field represents.
2. Specifications for the field.
3. Motivation for the inclusion of the field in our HSS.

Our analysis follows the subdivision of the HSS in classes. For examples on the use of the following fields, see Chapter 6, Section 6.6.

### 5.3.1 External Tutoring Management: ETM

This class represents any decisions on tutoring made outside Menon. The possibility to take such decisions into account adds to the flexibility of Menon as a tutorial manager. The *ETM* consists of three fields:

***Task*** $\leftarrow$ a string representing the task of the session, which is decided by the inter-session user model, where the decision about the right level of task to be set is made. For example, to prove that if $A \subseteq K(B)$, then $B \subseteq K(A)$.

***Strategy*** $\leftarrow$ a symbol representing the kind of teaching strategy chosen for the session. Currently, only the Socratic is implemented.

It sets the strategy to be used, which may be chosen by the learner or the teacher responsible for the learner.

***Undo*** $\leftarrow$ a symbol representing undoing the previous turn only, the previous step, or the previous proof.

It captures a decision by the student to take back part of the previous input. It pops the necessary values from various fields in LMCL, Tutoring History, and Tutorial Goal Status, but not the GMCL that represents the overall performance of the student within one hinting session.

[2]For definitions of dialogue moves see Chapter 4

### 5.3.2 Tutoring History: TH

*Tutoring history* represents all tutoring decisions up to the current Proof Step. It consists of four fields:

***Previous hint*** ← a list of symbols of hint categories already produced in the session. The hint categories it can hold are the ones defined in Chapter 4, such as `elicit-relevant-concept`, `elicit-subordinate-concept`, `give-away-inference-rule`, `give-away-antithesis`, `elicit-domain-technique`, etc.

It captures domain knowledge that students should know, as it has been previously given to them via a hint, as well as the kind of feedback already provided. It is different from the domain knowledge checks, as the student might still not be able to use the content of the hints given. Hints giving domain knowledge away increase the possibility of getting a correct answer, but do not affect domain knowledge, or mastery of it [Gertner *et al.*, 1998]. In order to counter-balance this effect, we check for use of domain knowledge independently, and we penalise the hinting session status for the number and kind of hints given (how much domain knowledge has been given away).

***Substrategy*** ← a symbol, the kind of current substrategy. It takes as values the substrategies defined in Chapter 6, that is:

1. performable step
2. diagnostics
3. recapitulation
4. misconception
5. spell-out-task
6. request-assistance
7. aligning
8. the meta-reasoning tasks

This field is used for providing information to the dialogue manager that is relevant to the discourse and dialogue structure (cf. Chapter 6).

***Previous substrategy*** ← a list of the kinds of substrategies used in the session. The substrategies are the same as in substrategy.

It helps to keep track of what has already been tried out, in order to chose the next strategy (cf. Chapter 6).

***Tutor task dialogue-move*** ← a list of the dialogue moves performed by the tutor in each turn of the session.

It allows monitoring the previous pedagogical feedback in order to decide on the next feedback. The elements of the list can be any of the tutor-task dialogue-moves defined in Chapter 4, namely *check-origin-problem*, *encourage*, *align*, *signal-dom-con-eval*, *prompt*, *hint*, as well as all conventional-task management and conventional-dialogue management dialogue moves.

### 5.3.3 Tutorial Goal Status: TGS

Here we capture information on how well the student follows our tutorial goals. We then fine-tune our feedback based on it. The *TGS* consists of three sub-classes, i.e., instructional points, pragmatic information, proof status, and of the following three fields:

***Student task dialogue-move*** ← a super-class of the dialogue-moves that the student performs at the tutoring-task level[3]. The student task-dialogue-moves held in these fields are mutually exclusive.

Let us investigate the different possible instances of the super-class. An analysis of these instances with regard to the dialogue model and the obligations deriving from the dialogue model can be found in Chapter 4.

1. *Resign* ← a symbol representing that a *resign* has occurred.

   *Resign* is an utterance with which students indicate that they give up the proof-task, as opposed to asking for specific help in an attempt to solve the task (cf. Chapter 4). Therefore, we use it as an indicator of low motivation, demonstrated as lack of effort, that needs to be treated differently from wrong answers [de Vicente and Pain, 1998; du Boulay and Luckin, 2001]. Moreover, it has been observed that humans behave differently with computers than with other humans [Shechtman and Horowitz, 2003]. For our purposes, this might translate into a reluctance to cooperate, in which case *resign* can be an indication of such a reluctance.

2. *Time-out* ← a symbol representing that a *time-out* has occurred.

   It shows that students have some difficulty. At best, they are taking too long to answer. Otherwise, they do not know what to do at all. It is better than resign, as students might be trying for the answer but taking too long and be lost [Lim and Moore, 2002].

3. *Request-assistance* ← a symbol representing that a *request-assistance* has occurred.

   *Request-assistance* concerns only specific information requests, e.g., about concepts. We represent Request-assistance in the HSS to be able to provide

[3] *Domain-contribution* does not belong here, as it is not an object of tutoring task, but of proof task.

the assistance requested, depending on other considerations (cf. Chapter 6). Request-assistance is represented through the assistance requested, as follows.

(a) Assistance-requested ← a symbol representing the kind of assistance that the student requests. It is a super-class of the possible kinds of assistance requested, that is some piece of domain knowledge. Its instances are, (i) some of the domain knowledge represented in passive hint categories, e.g., which is the Relevant Concept, which is the Specific Method, why is this the Proof Step, a definition that constitutes Basic Knowledge for the step etc., (ii) that the domain information requested belongs to a different theory than the one in the current proof task, or (iii) that the domain information requested is casted as irrelevant to the session and the task.

(b) Previous assistance-requested ← a symbol, the kind of assistance requested that has already been dealt with in the session. The values are the same as in assistance-requested.

It makes it possible to reason about how to handle a request for assistance if it is of the same type as a previous request for assistance.

4. *Request-evaluation* ← a symbol representing that a *request-evaluation* has occurred.

   *Request-evaluation* is an utterance with which the student inquires explicitly about his progress. For example “How am I doing?” requests an evaluation of the overall performance, and “Was that right?” requests an evaluation of the current student answer, be that a whole proof step, or part of it (cf. Chapter 4). In both cases, the fact that the student is requesting an evaluation of her performance can be a sign of demotivation and need for reassurance.

***Previous student task dialogue-move*** ← a super-class of the dialogue-moves that the student performed in the previous turn at the tutoring-task level. The instances of the class and motivation are as in student task-dialogue-move.

***Domain-contribution*** ← a symbol representing that a *domain-contribution* has occurred. Because it pertains to the proof task, it demands the corresponding manipulation of it by the tutorial manager. It is always assigned a domain-contribution category.

Domain-contribution category ← a super-category of the categories correct, wrong etc. that we will now define and that represent the category of the *domain-contribution*.

This is the most local evaluation of the student’s input, which is also the most important for the decision on feedback [Freedman *et al.*, 1998]. Although it is not within the scope of this thesis to formally define a categorisation scheme for the domain-contribution category, we present here a partial formalisation, which was developed in the context of the DIALOG project [Tsovaltzi and Fiedler, 2003a]. The scheme was put to test in the Wizard-of-Oz experiment, as the human wizard used it to annotate the student input with a domain-contribution category [Wolska *et al.*, 2004; Benzmüller *et al.*, 2003b].

The student's answer is evaluated by use of an expected answer.

**Expected answer:** The *expected answer* is the proof step that is expected next according to the formal proof that the system has chosen for the problem at hand.

We want to make use of the students' own reasoning in helping them with the task and avoid super-imposing a particular solution. OMEGA models that by trying to match the student's answer to a proof step in one of the valid proof developments derived from previous student attempt (cf. Section 3.3). This we call *proof-step matching*. At the cognitive level, proof-step matching is a means of promoting implicit learning, motivation, and schema acquisition in general (cf. Chapter 2 and [Tsovaltzi and Fiedler, 2003a]).

We also define the expected sub-answer.

**Expected sub-answer:** The *expected sub-answer* is a part of the expected answer that is in the focus of the tutoring, i.e., the previous active hint is requesting the domain information involved in it.

**Parts of Answers and Over-Answering** We now look at the relevant units for the categorisation of the student answer by the domain reasoners.

**Part:** A *part* is a premise, the conclusion or the inference rule of a proof step. The two former are mathematical formulae and must be explicitly mentioned for the proof step to be complete. The Rule of Inference can either be referred to by name, or it can be represented as a formula itself. This is up to the student.

**Subpart:** A *subpart* is any of the instructional points defined in Chapter 3 except the ones that are defined as parts.

Over-answering (accurate or inaccurate) may be considered as several distinct answers. That is, if the student's answer has more proof steps than one, the steps are considered as multiple answers. The categorisation would normally be applied to them separately. Nevertheless, there are cases where the order of the presentation of the multiple answers is crucial. For example, a correct answer that is inferred from a previous wrong answer cannot be counted, since it would have to follow from a wrong premise.

The predicates complete and accurate are also necessary for the categorisation of the student answer in the Wizard-of-Oz experiment:

**Complete:** An answer is *complete* if and only if all parts of the expected answer are mentioned.

**Accurate:** A part or a subpart of an answer is *accurate* if and only if the propositional content of the part or subpart is the expected one.

**Completeness** We distinguish between getting the expected domain object right and instantiating it correctly. The latter does not follow from the former. Completeness is a two-value relation. It refers to the presence of the object but not to its correct instantiation. In other words, a place holder for an expected object in the answer is enough for attributing completeness, no matter if the object itself is the expected one. That issue is dealt with by accuracy.

**Accuracy** Accuracy refers to the appropriateness of a part or subpart in the student answer with respect to the expected one. A part or subpart is accurate if and only if it is the exact expected one. Contrary to completeness, accuracy is fully dependent on the domain ontology.

Based on this analysis, a complete and accurate proof step is always a correct *domain-contribution*. However, it is possible that the *domain-contribution* is considered correct, although the step is not completed. Namely when the student input is complete and accurate with respect to the content of an immediately preceding hint that elicits some specific part or subpart of the proof step. For example, when the preceding hint elicits the Rule of Inference and the student's domain-contribution provides the accurate Rule of Inference for the step. This is a correct *domain-contribution* despite the fact that in terms of the proof step completion it would be considered incomplete-accurate. Therefore, the same input when no hint precedes should be categorised as a partial-answer *domain-contribution* (For examples, cf. Chapter 6, Section 6.6).

We made the following enhancements to the original categorisation scheme, based on the data from the Wizard-of-Oz experiment.

First, for the purposes of the original procedure used in the Wizard-of-Oz experiments, we counted the categories complete-partially-accurate, incomplete-partially-accurate, and complete-inaccurate as wrong. However, after the experiment, we represent complete-inaccurate explicitly.

Second, the analysis of our experimental data also showed that it is more useful to define in more detail which parts are missing or which are incorrect. Therefore, we collapse incomplete-accurate, incomplete-partially-accurate to partial-answer and instead we enriched our instructional points and use them as meaningful parts that are pedagogically interesting and are represented in the HSS (cf. Chapter 3). These are now the subparts of the expected answer that we consider in the student's contribution for categorising it. The category complete-partially-accurate was kept only for informing the student of the quality of the answer, but it is otherwise also treated as a partial-answer.

Third, the old scheme used accuracy as a two-value relation in the same way as completeness. The data revealed additional interesting values of accuracy, apart from accurate and inaccurate. Therefore, we now also consider intermediate categories useful. All the categories below, apart from correct, partial-answer, complete-inaccurate, and wrong, are alternative values of accuracy.

The domain information classes and the instructional points used by them also follow the accuracy vs. completeness dichotomy. The questions that the domain information classes consider for every instructional point are:

1. Is the instructional point present?
2. Is the present instructional point accurate?
3. If the instructional point is not accurate, does it bear a domain relation to the expected subpart?

The difference between the instructional points that we define and other possible subparts of an expected answer is that instructional points are meaningful parts used for *schema* acquisition. Therefore, hinting concentrates on them.

We now explore the instances of the domain-contribution category and how they are important to the HSS.

1. Correct ← a *domain-contribution* that is both complete and accurate.
2. Partial-answer ← this may be one of the following:
   (a) Incomplete-accurate ← a *domain-contribution* that is incomplete, but all parts or subparts that are present in it are accurate.
   (b) Incomplete-partially-accurate ← a *domain-contribution* that is incomplete and some of the parts or subparts in it are inaccurate.
3. Complete-partially-accurate ← a *domain-contribution* that is complete, but some parts in it are inaccurate.
4. Complete-inaccurate ← a *domain-contribution* that is complete, but all parts in it are inaccurate.
5. Wrong ← a *domain-contribution* that is both incomplete and inaccurate.

   From this point on, all categories deal with values of accuracy other than inaccurate and accurate.
6. Unknown ← a *domain-contribution* that cannot be categorised otherwise, although the linguistic content is parsed.
7. Misconception ← a *domain-contribution* that reveals a problem with a part or subpart that affects not only the current proof step and proof, but potentially the understanding of the whole domain. Such a *domain-contribution* may at the same time have different degrees of correctness and completeness, but the fact that it is a misconception overrides for the purposes of tutoring any other categories that might apply to it.

   The misconceptions that we currently model comprise the following:

   (a) Any instance when students do not understand that a near-miss is a wrong use of the domain relation involved, after receiving a hint on it. The misconception is the wrong understanding of the domain relation (cf. Chapter 3 and Appendix A).

(b) A *domain-contribution* that is classified as a hypotaxis, which is not corrected after the relevant hint.

(c) A *domain-contribution* that is classified as a primitive, which is not corrected after the relevant hint.

(d) step-size if students do not realise that they are missing intermediate steps, and they do not just consider them trivial, after receiving a hint about it.

Once the NL analysis has recognised the *domain-contribution* and the fact that the student has not answered correctly the hint produced as reaction to the *domain-contribution* each time, Menon can identify these types of misconception. The misconception types are needed as parameters for the NL generator to provide the right explanation.

8. Missing basic knowledge $\leftarrow$ a *domain-contribution* that shows that the student does not know some defined constants, i.e., declarative knowledge like definitions of concepts, definition of Rules of Inference etc.[4]. The missing basic knowledge relates to a part or subpart of the expected answer.

   These definitions are included in the lesson material that the student reads as preparation for the interactive phase of tutoring investigated in Chapter 6. The basic knowledge is always relevant to the Rule of Inference to be applied, as it is either the Rule of Inference itself or it is related to the Relevant or the Subordinate Concept whose role is to assist the student in finding the right Rule of Inference.

9. Step-size $\leftarrow$ a *domain-contribution* that does not follow from the previous steps without proving it. In other words, intermediate steps are required.

   Step-size is handled by the proof manager, as part of the issue of granularity [Schiller *et al.*, 2006] and is also relevant to over-answering.

   We aim at tutoring correct deduction, so students must learn to justify what they do [Wu, 2001].

10. Irrelevant $\leftarrow$ a *domain-contribution* that is correct (the theorem prover will indeed prove it), but does not contribute anything to the solution of the task at hand.

    For motivation and accuracy reasons, we inform the student that the mathematical statement is correct, but it is not useful for this proof.

    Missing basic knowledge and misconceptions are different from irrelevant *domain-contributions* (normally questions) that are ignored completely, because they would just sidetrack too much from the task. Such knowledge involves what is in principle part of some other theory, or not directly associated with what is being taught. In the domain of set theory proofs, everything that belongs to logic falls under this category.

[4] Missing basic knowledge is different from primitive concept, which is a misconception category.

11. Ill formed ← a *domain-contribution* with a syntactic mistake in a part or subpart of the expected answer.

    It is basically related to the Substitution of some inference rule. For instance, the student misses a bracket, or wants to say $B$ and says $b$, that is the student wants to refer to a set and instead refers to the elements of the set as in $(x \in b)$, where $B$ denotes a set whereas $b$ denotes an element [Horacek and Wolska, 2006a].

    We represent syntactic errors since accuracy in mathematics is important [Wu, 2001].

12. Wrong linguistic-term ← a *domain-contribution* with terminology errors relating to a part or subpart of the expected answer.

    For example, the student wants to say "element of" and says "contained in", or "the union of sets" and says "both sets together" [Horacek and Wolska, 2006b]. The use of correct terminology is one of our tutorial goals and another aspect of accuracy in mathematics [Wu, 1996].

13. Near-miss ← a partial answer *domain-contribution* that differs from the expected answer in one part or subpart. This part or subpart is not the expected one, but bears one of the following Domain Relations to the expected one: antithesis, duality, conversion, specialisation, generalisation. This is a subset of the domain relations defined in our domain ontology. Otherwise, a near-miss is a complete partially-accurate proof step, whose inaccurate part or subpart is a wrong linguistic-term or an ill formed one.

    In general, a near-miss is an answer that differs from the expected answer only by one concept[5]. This is different from a *domain-contribution* where the only correct element is the concept that bears a Domain Relation to the expected concept. That would not be a near-miss, but a wrong answer. The HSS is affected differently and so is the choice of the procedure behaviour, namely to carry on hinting or not.

    The motivation behind this category is that it captures the students' existing mental structures and thus may help the satisfaction and the sense of achievement of the students.

***Previous domain-contribution*** ← the previous *domain-contribution* performed by the student.

It is analogous to the current *domain-contribution* and hence is also characterised by one of the domain-contribution categories.

#### 5.3.3.1 Instructional Points

We represent instructional points, which are required for building both the proof and also mental schemata for constructing the proof. We now list our

[5]This is the definition given by our human tutor's in the Wizard-of-Oz experiments, who made use of this distinction.

instructional points.

***Relevant Concept used*** ← a Boolean representing whether the student knows/has used the Relevant Concept.

***Subordinate Concept used*** ← a Boolean representing whether the student knows/ has used the Subordinate Concept.

***Domain Relation used*** ← a symbol, the kind of Domain Relation used, if any at all. It captures a Boolean representing whether the student knows or has used a concept that stands in a Domain Relation with the required concept e.g., antithesis, duality etc. It is relevant when the expected subpart is this concept only.

***Rule of Inference used*** ← a Boolean representing whether the student knows/has used the Rule of Inference.

***Inverse rule used*** ← a Boolean representing whether the student has used the inverse of the Rule of Inference required.

It refers to the relation *inversion*, defined in our domain ontology. It captures the fact that the student might be on the right track, but made a mistake regarding the direction of the rule application[6].

***Technique used*** ← a Boolean representing whether the student knows/has used the needed technique.

***Starting Point used*** ← a Boolean representing whether the student knows where the reasoning should start from.

If students know the premise and the conclusion, or if they have stated that they have to start from finding out what the premise and the conclusion are, then the value of the field is true.

***Premise used*** ← a Boolean representing whether the student knows/has used the premise of the current step.

***Conclusion used*** ← a Boolean representing whether the student knows/has used the conclusion of the current step.

***Directness used*** ← a Boolean representing whether the student knows whether the proof is direct or an indirect.

***Direction used*** ← a Boolean representing whether the student knows which proof direction to use (forward vs. backward steps).

[6]It was observed and used as a hint in the Wizard-of-Oz experiments.

#### 5.3.3.2 Pragmatic Information

These fields concern information about the Proof Step that is different from information relevant to the domain-contribution category. Therefore, a *domain-contribution* may be annotated for the category at the conceptual level and at the same time for one of the parameters listed here for the pragmatic characteristics available. Pragmatic information may be, for example, the number of subparts that are necessary for an answer to be complete, or that there is a previous occurance of some domain information in the course of the same tutoring session, or that there is some mistake in terminology, etc.

***Different theory*** ← a Boolean representing whether some information belongs to a different theory than the one in current the hinting session.

At the moment, it takes only one value: next step. It allows the tutor to stay within the limit of the theory intended for tutoring.

***Same Domain Information*** ← a symbol, the kind of domain information that is potentially the same between two different proof steps.

The values are the same as those in assistance requested (apart from irrelevant assistance requested), representing all possible instructional points of the performable step and the meta-reasoning. An example is that the Rule of Inference for two different steps in the same proof is the same.

It allows references to previous explanations of the same piece of information for more structure in tutoring.

***Ordered list available*** ← a Boolean representing whether the expected answer is an ordered list as presented in the lesson material. It means that there is a list of points that need to be made in the expected answer in a specific order (e.g., the cases in a case split), and the student's *domain-contribution* is missing out at least one point.

***Unordered list available*** ← a Boolean representing whether the student is missing some or all elements of a list that is presented in the lesson material as an unordered list. The *domain-contribution* can be wrong or partial-answer (incomplete-accurate), and the points in the expected answer do not have to occur in any particular order.

#### 5.3.3.3 Proof Status: PS

The *proof status* provides basic information that influences the updating of the MCL, as well as the continuation of the tutoring task.

***Proof Step completed*** ← a Boolean representing whether the Proof Step is completed.

***Proof completed*** ← a Boolean representing whether the proof is completed.

### 5.3.4 Motivation and Cognitive Load: MCL

The *MCL* is split into global and local. The first represents the performance of the student throughout a hinting session. The second represents the performance during one proof step only.

#### 5.3.4.1 Global Motivation and Cognitive Load: GMCL

The *GMCL* represents the student performance in the hinting session as a whole. All counts below contribute to this representation either directly, as they show the level of the students' understanding and cognitive load, or more indirectly, as they constitute situations in which students might be demotivated or frustrated. The rationale here is simply that low performance often demotivates students. The GMCL consists of thirteen counts.

The GMCL informs the decision on the substrategy to be used and whether information should be elicited or given away.

***Proof-step-count*** ← the number of performable steps so far performed by the student.

It is relevant to understanding and frustration. The number of proof steps performed is a measure of cognitive load, as it indicates how slow or fast the student is moving in the task [Sweller and Chandler, 1991].

***Unknown-count*** ← the number of unknown (unclassifiable) *domain-contributions*.

It is an indication of a low performance.

***misconception-count*** ← the number of misconceptions.

This is not only relevant to the current hinting session, but does also show that the student has a more general problem, which also affects the current performance. Therefore, it is included in the GMCL.

***Missing-basic-knowledge-count*** ← the number of missing-basic-knowledge *domain-contributions*.

It shows that the student does not possess enough domain knowledge of that presupposed by the current learning goal for the session.

***Request-assistance-count*** ← the number of *request-assistance* dialogue moves.

It shows inability to proceed and possibly low confidence.

***Resign-count*** ← the number of *resign* dialogue moves.

It shows low confidence, dissatisfaction, and lack of motivation.

***Wrong-count*** $\leftarrow$ the number of wrong *domain-contributions*.
It indicates a problem in understanding, as do the next five fields.

***Step-size-count*** $\leftarrow$ the number of step-size *domain-contributions*.

***Irrelevant-count*** $\leftarrow$ the number of irrelevant *domain-contributions*.

***Ill-formed-count*** $\leftarrow$ the number of ill-formed *domain-contributions*.

***Wrong-linguistic-term-count*** $\leftarrow$ the number of wrong-linguistic-term *domain-contributions*.

***Time-out-count*** $\leftarrow$ the number of time-outs.

***Domain-contribution-count*** $\leftarrow$ the number of *domain-contributions*.
By representing the domain-contribution-count, we can take into account the non-wrong (as opposed to only correct) answers so far in the session, which also shows the extent to which the student is following the hints, or the task in general. We do that since in our model it is not so much wrong answers that are decisive as the non-wrong *domain-contributions*. This means that as long as the student gets something right, we carry on hinting. The domain-contribution-count is also used to allow for a slow start, which is common for Socratic-oriented teaching strategies [Mathews *et al.*, 1989].

We use the GMCL counts in the following way. We add all previous counts and divide by the domain-contribution-count. We call the derived aggregate the *Global Motivation and Cognitive Load Aggregate* (GMCLA). If GMCLA $\leq$ 0.5, then the *domain-contributions* that show poor performance are lower than the mean of the *domain-contributions*. This allows us to make the following inferences, which guide our Socratic strategy:

1. GMCLA $\leq 0.75$: A very good performance.
2. GMCLA $\leq 0.5$: A performance above average.
3. GMCLA $> 0.5$: A performance below average.
4. GMCLA $\geq 0.3$: A very poor performance.

These inferences are intuitive, but it is still an open empirical question whether they are accurate or not.

#### 5.3.4.2 Local Motivation and Cognitive Load: LMCL

The *LMCL* applies to one proof step only. As with the GMCL, it is used to capture situations in which a student might be demotivated or frustrated and situations which may indicate that unnecessary cognitive load is imposed upon the student. However, the LMCL fields carry more weight with regard to the feedback decisions, exactly because they relate to the very local performance [Freedman *et al.*, 1998]. Therefore, these fields are inputted explicitly to the Socratic procedure. The LMCL consists of four counts:

***Local wrong-count*** ← the number of wrong *domain-contributions* during a proof step.

***Local correct-count*** ← the number of correct *domain-contributions* during a proof step.

***Local domain-contribution-count*** ← the number of *domain-contributions* during a proof step.

***Local hint-count*** ← the number of hints during a proof step.

As with the *proof-step-count*, the number of hints provided is a measure of cognitive load, as it shows if the student is fast in following the hints.

For Menon's complete input representation language, see Appendix B.

## 5.4 External Input vs. Internal Fields

In this section we look at which of the HSS fields are input to Menon and which are maintained by Menon. We also provide indications on which modules in the dialogue manager architecture in Chapter 1 provide the information for the respective fields.

### 5.4.1 External Input

Any HSS field that depends on domain reasoning, or linguistic and dialogue analysis, as well as user decisions represented in the HSS constitutes input to Menon. In particular, the input is:

1. all fields of the External Tutoring Management (ETM), as they are user choices to be communicated via the dialogue manager
2. the two Proof Status fields, but only when the completion is done by the last *domain-contribution* performed by the student (in that case, both fields relate to the proof task represented in proof manager, which reasons about the completion of the step and proof).
3. the recognition of the instructional points in a *domain-contribution* that involves domain reasoning (cf. Chapter 3)

4. the student task-dialogue-move and the domain-contribution with its category each time, both of which pertain to the analysis of the input (the domain-contribution category requires additional domain reasoning, which is obvious when looking at the formalisations presented in Section 5.3.3).

### 5.4.2 Fields Maintained Inside Menon

Fields that are internally maintained consist of everything in the HSS that is related to previous tutor output, pedagogical interpretations of the student input, or previous values of fields that are kept for pedagogical interpretations. These include:

1. all fields of the Tutoring History

2. instructional points, when those are given away by a hint, as opposed to them being included in the student's input

3. all of the counts of MCL, as well as the dynamic value of the GMCLA

4. the Proof Status values, Proof Step completed and proof completed if they are completed by the hint produced by the tutor

5. previous assistance-requested, previous student task-dialogue-move, and previous domain-contribution

These fields are updated internally.

Finally, all HSS fields are reset as required by the teaching strategy (cf. Chapter 6).

## 5.5 Types of Mistakes

To provide an overview of the kinds of mistakes that students might make, the HSS represents the following mistakes, detected during the lignuistic and domain analysis of the student input:

1. Mistakes that capture the result of the comparison of the expected answer to the student *domain-contribution*, or the pedagogical information available for the expected answer.

2. Mistakes that manifest a more general problem with the domain knowledge.

The first type of mistakes are represented in Menon by the instructional points and their analysis, as well as by the different subcategories of near-miss (see above), the pedagogical information that we look for in the answer, and the tracking of other relations defined in the domain ontology (currently the inverse rule). Since our instructional points also cover the meta-reasoning for the step, we also cover mistakes relevant to the meta-reasoning.

We represent the more general domain mistakes mainly by the categories misconception and missing basic knowledge. Under this type, we also consider mistakes that are not part of the domain theory under investigation, in our case the mathematical theory being taught. We do not treat such mistakes in the same way as we treat ones in the current mathematical theory, but we do acknowledge them (cf. Chapter 6).

In general, different kinds of mistakes are treated via the choice of hint dimensions and what they capture (cf. Chapter 4), as there is naturally a correspondence between the hints provided and the kind of mistakes in the student input.

## 5.6 Conclusion

The task-level dialogue-moves, which are used by the HSS, require formalisation at the level of the linguistic input. Although a first attempt has been made towards this end (cf. Chapter 4 and [Tsovaltzi and Karagjosova, 2004]) this is the task of the natural language analysis, which is not within the scope of this thesis. This also holds for other HSS fields where linguistic information is relevant, e.g., wrong linguistic-term, ill formed etc. Linguistic research in this direction might disclose aspects of the student input that are not covered by the existing categories available to the tutorial manager. This is always a possibility, as both the student input recognition and the problem of providing appropriate feedback are AI complete problems.

# Chapter 6

# Socratic Teaching-Strategy

## 6.1 Introduction

This chapter analyses the teaching strategy. It provides an overview of the architecture of Menon, explains the Socratic teaching strategy that we implement and how this is realised in Menon. Moreover, the chapter details the outcome of the Socratic teaching strategy, based on the relevant hinting-session status, and motivates the choices of the strategy from a pedagogical perspective. We include an explanation of the output for specific domain-contribution categories, as well as a description for subtasks, subdialogues, and dialogue moves produced by the tutor.

## 6.2 Overview of Menon

Before we look at Menon's architecture, let us explain its general design. *Strategies* are generic teaching methods that are based on a pedagogical teaching model implementing it. Strategies consist of generic pedagogical feedback and substrategies. *Substrategies* are the different methods, which assist in realising the strategy and are applied for different hinting situations. In other words, they define the top level strategy. Substrategies can be called recursively and within one another. They are divided into subtasks and subdialogues based on their primary function (cf. Section 6.5). *Class subtasks* correspond to the group of hints relating to one instructional point. They pick the appropriate hint from the group of hints that address a main instructional point (cf. Chapter 3). Hints and other tutor-task dialogue-moves are the basic units of the teaching strategy concerned with cognitive feedback.

We now describe the architecture of Menon (right-hand side of Figure 6.1) and how the task dialogue-moves and the hint definitions (cf. Chapter 4), as well as the HSS (cf. Chapter 5) are used. In order to set Menon in context, we also give at places brief descriptions of a potential tutorial-dialogue manager, in which Menon constitutes the tutorial manager module. The tutorial-dialogue manager

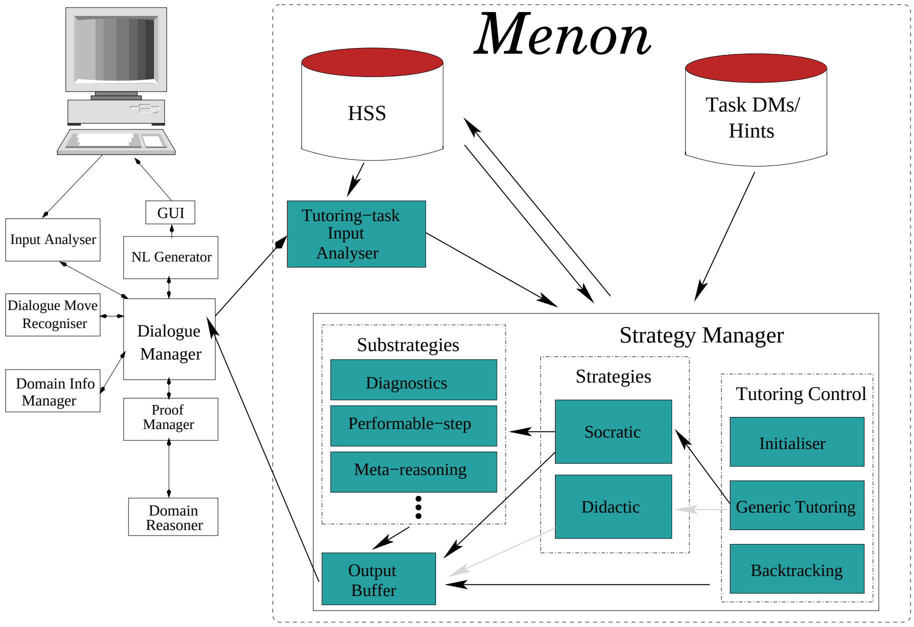


Figure 6.1: Menon architecture

we look at (left-hand side of Figure 6.1) is that of the DIALOG project [Buckley and Benzmüller, 2005; 2006]. In the context of such a tutorial-dialogue manager, `Menon` is responsible for the production of all tutoring feedback. This involves choosing the appropriate type of feedback, making hint-dimension and substrategy choices, and providing explicit motivation.

`Menon` was implemented in Common Lisp on a Linux workstation.

We will now examine the individual submodules of `Menon`, illustrated in 6.1. Cylinders represent data bases and boxes procedures. Arrows represent the flow of information to the submodule where the arrow points to. The input to `Menon` is the analysed student input (written in the HSS). The output consists of dialogue moves and their potential parameters and is organised into general pedagogical feedback, conventional management and main feedback to mirror the analysis presented in Chapter 4. Figure 6.2 shows an instance of the output Lisp class.

## 6.3 Tutoring Control

TUTORING CONTROL (cf. Figure 6.1) takes care of control issues and any tutoring behaviour that is independent of strategy choice. More specifically, first

```
(#< IO+OUTPUT-CLASS
:genFeedback (#<DM+TUTORTASKDMS-CLASS :kind ENCOURAGE :dm-
par NIL>
                #<DM+DOMCONEVAL-CLASS :kind SIGNAL-PA :dmpar
NIL>)

   :conMan (#<DM+CONTASKMANDMS-CLASS :kind INITIATESUBTASK
:dmpar #<SS+METAREAS-CLASS :mode METAREAS :type PROOFSTEPM>>)

   :mainFeedback (#<DM+CONCEPTUAL-HINTS :category PREMCONC
:elicitation ACTIVE :dom-know PROOFSTEP :infRole METAREAS :dmpar
NIL>)
>)
>)
```

Figure 6.2: An instance of `Menon`'s output class

it initialises the module. Second, it initiates the Tutoring-Task Input Analyser, which reads the input to `Menon`. This analysed input consists of information, which pertains to the analysis part of a dialogue system, and is dependent on either natural language, dialogue, or proof-task analysis. The input updates the HSS fields that hold external information, that is the Tutorial Goal Status and the External Tutoring Management (cf. Chapter 5). This updating of the HSS is done recursively after every output by `Menon` and input to it. Task DMs and Hints is also loaded.

Third, GENERIC TUTORING is called, which takes care of issues that are commonplace among different teaching strategies and sets the strategy of choice for the tutoring session. It initiates the dialogue and then the task upon the student's response[1]. It accepts any correct answers and calls the subtask ***recapitulation*** if the answer completes the proof, to recapitulate the steps of the proof. If further or alternative feedback is necessary, it calls STRATEGIES (cf. Section 6.4). Finally, if the student takes back a turn, a step or decides to start with the new proof, the function BACKTRACKING is called. This down-dates the HSS, based on what the student has taken back.

Every output produced by the different modules in `Menon` is combined in the Output Buffer and sent to the Dialogue Manager. Any mathematical instantiations for the particular proof step and the domain knowledge included in the hints takes place in the domain reasoners Proof Manager and Math KB. Any non-tutoring-related output can be generated for the student while `Menon` processes the new input and before the tutoring feedback is produced. All out-

[1] `Menon` assumes an external student model, which is responsible for choosing the right level of task for each session.

put information concerning tutoring, natural language, dialogue and discourse is finally sent to the NL Generator. There, it is collected for the sentence level realisation of the feedback to the student while `Menon` is expecting the next input.

## 6.4 Strategies: The Socratic Teaching Strategy

STRATEGIES may include any desired teaching strategy and the functionality of choosing among strategies is provided. We implement our Socratic teaching strategy in *`Socratic`*. It is the implementation of our pedagogical model (cf. Chapter 2).

Hinting starts if the student does anything apart from performing the next correct performable step. However, once the hinting has started, other answers count as correct, as well. That is, whatever matches the content of the passive hints counts as a correct answer to the corresponding active hint (cf. Chapter 4). The more non-correct answers the student gives, the more likely it is that the tutor[2] will switch into a more guiding substrategy in order to prevent both giving the answer away and frustration on the part of the student, always depending on the general hinting situation as well. We call this *`spell-out-task`* (cf. Section 6.5).

The tutoring feedback is divided into (i) production of appropriate hints based on the formalisation in Chapter 4, (ii) pedagogically-motivated generic output. The theory of discourse obligations [Matheson *et al.*, 2000], adapted for the genre of tutorial dialogues [Tsovaltzi and Matheson, 2002], along with motivation-theory considerations inform the choice of pedagogical feedback other than hints (cf. Chapter 4).

Hint categories are selected through the choice of the right cognitive function in every dimension (see Chapter 4), represented either in the choice of subtask for pragmatic vs. conceptual and performable step vs. meta-reasoning, or in the choice of class subtasks and instructional point (Domain Relation vs. Domain Object vs. Rule of Inference vs. Substitution vs. Proof Step). The active vs. passive decision is made inside the subtasks.

Inside *`Socratic`*, SOCRATIC GENERIC (cf. Appendix E) is a control function, which does the following:

1. it produces any applicable generic Socratic-feedback, e.g., motivational feedback
2. it calls the appropriate substrategy, e.g., *`performable-step`*
3. it makes the choice between pragmatic vs. conceptual in the Problem-Referential Perspective dimension

Generic Socratic-feedback comprises, (i) encouraging the student, (ii) signalling the evaluation of the student's *domain-contribution*, (iii) as well as signalling the closing and initiation of substrategies.

[2] "Tutor" will be used from now on to mean the tutorial manager.

The decision on calling a substrategy involves if a subtask or a subdialogue is appropriate for the current hinting situation (cf. Section 6.4.1) and which subdialogue or subtask is appropriate. The subtasks that we implement are: *pragmatic*, *performable-step*, *meta-reasoning*, *spell-out-task*, *request-assistance*, *explain-misconception*, *aligning*, *near-miss* and *recapitulation*. *Performable-step* and *meta-reasoning* are subdivided in *class subtasks*. We also implement the *diagnostics* subdialogue, which is a means of pinpointing the problem, in order to pick a subtask for dealing with it (cf. Section 6.5).

The choice between pragmatic vs. conceptual is made by calling *pragmatic*, or *conceptual*. *Pragmatic*, chooses the appropriate pragmatic hint, when the analysis which is passed on to Menon detects that the student input bears interesting non-content characteristics (cf. Section 6.5.2). *Conceptual* is a control strategy for choosing among the various subtasks that produce conceptual hints. Hence, *conceptual* controls all other subtasks involved in hinting, apart from the *pragmatic* subtask, which is orthogonal to it. *Conceptual* does though use a pragmatic hint, that is the hint that instructs the student to read the preparation material again, exactly at the point when the conceptual hints appear not to work and the whole hinting process starts afresh.

The principles of CONCEPTUAL (cf. Appendix E) where derived from the informal analysis of the BEE [Moore, 2000] and DIALOG [Wolska *et al.*, 2004] corpora, from previous work on hinting [Hume, 1995] and from the guidelines of our tutoring model (cf. Chapter 2). These principles are summarised as follows.

1. Up to three wrong *domain-contributions* from the student and two hints from the system, it calls the *performable-step* subtask (cf. Section 6.5). Although Gregory Hume [Hume, 1995] found that two is the right number of wrong *domain-contributions*, we increased it by one, as two wrong answers seemed too few in our experiments, where the human wizard felt that the students were not given enough opportunity to find the answer themselves, and based on our analysis of the BEE corpus.

2. Up to five hints, when the current *domain-contribution* is wrong or irrelevant, but at least half of the *domain-contributions* are correct or when the current and previous *domain-contributions* are correct, it still calls *performable-step*. In both cases, the student seems to be following hints reasonably. Therefore, there is no reason to stop, as the motivation levels should be high enough.

3. If less than half the *domain-contributions* are correct, then the tutor gives the pending answer away, as well as the current proof step with its meta-reasoning, to prevent frustrating the student.

4. For any other domain-contribution category, the *spell-out-task* subtask is called, which is far more assistive in that it gives answers away more easily to avoid demotivating the student. If the student gives more correct answers again after switching to the subtask (*spell-out-task*), this is a

sign that the student is following the proof again. Thus, the tutor need not explain anymore and the procedure switches back to the standard `Socratic` strategy.

`Performable-step` (cf.Appendix E) makes three choices:

(a) how informative the hint should be, that is which instructional point needs to be addressed

(b) if the hint will be active or passive, as defined in Chapter 4

(c) or if one of the `meta-reasoning` subtasks should be called (cf. Section 6.5.3)

5. Finally, if the hints become more than five, we only call the subtask `spell-out-task`, as long as the *domain-contributions* are correct. In any other case, we interrupt the hinting session and ask the student to read the lesson preparations material. This behaviour was observed in the BEE corpus. We adopted it because continuing hinting when the student does not follow well would probably just result in further demotivation, or in the student getting the answers without understanding them (cf. Chapter 2).

### 6.4.1 Socratic Output for Hinting Situations

In this section we look into the more detailed output, departing from the abstract strategy choices. We show the output that `Socratic` produces based on specific HSS fields, which constitutes the Socratic teaching strategy [Collins and Stevens, 1982]. This presentation aims at allowing the reader to see directly how the HSS fields influence the output. Therefore, it does not reflect the implementation structure of the feedback, as the previous section did. Examples are provided where clarification is necessary. The examples of this chapter are based on the current output of `Menon` and the NL phrasing of the student input and the tutor output are constructed examples, unless otherwise indicated. **T** stands for tutor and it represents the output by `Menon`. **S** stands for student. Before each separate output, the type with any parameters of the output is shown in brackets. Wherever original text from the corpus is used, we provide the German version and an English translation, as well as the participant's code. Wherever "K" is used it stands for "complement" and "P" for "powerset".

Virtually all hints can be used at any point if students themselves take the task initiative and ask something that requires a specific kind of hint. This is implemented in subtask `request-assistance` (cf. Section 6.5.5). The following constitutes output of the strategy manager when the task initiative is maintained by the tutoring system.

#### 6.4.1.1 External Tutoring Management

The external tutoring management in general allows setting values of `Menon` externally if necessary.

***Task*** It is used as a parameter in the dialogue-move *initiate-task*, to set the task for the session.

***Strategy*** It allows setting the teaching strategy externally. It is provided for engineering purposes.

***Undo*** It allows students to take back a turn, step or even start the proof from scratch. In other words, it allows students to do something wrong and change it, or even to start an attempt but finally move to a solution they are more comfortable with. This is all part of the non-goal oriented teaching model. (cf. Example 6.6.4) The following cases are implemented:

1. If the student takes back the whole proof, and in effect starts a new proof, we reset the HSS.
2. If the student takes back a step, we reset the LMCL, the tutoring history, and the tutorial goal status, which hold values relevant to the current step.
3. If the student takes back a turn, we reset the local hint-count and reload the values of the HSS before the last turn.

#### 6.4.1.2 Tutoring History

***Previous hint*** It assists the choice of output based on domain information given away in the previous turn. It helps decide first on the focus of the feedback, and then between eliciting or giving away information (cf. Example 6.6.2, **T10-T12**) and which exactly that domain information shall be. For example, if the hint `give-away-inference-rule` fails, then the procedure will choose a `substitution-class` hint, calling the corresponding function. Whether the student possesses the knowledge of the content of the most informative hint in every class is the condition of termination of the corresponding substrategy.

***Substrategy*** It is used to signal the initialisation of a new subtask to strucure the tutoring. Tutors in both the BE&E corpus did that consistently. See Example 6.6.3, eg. **T2, T4, T5, T7**.

***Previous substrategy*** It is used to keep track of the substrategies applied in order to maintain a consistent approach to the treatment of the student's input. For instance, when a misconception is being treated, the same substrategy (*`explain-misconception`*) is called until there is nothing more to do for the misconception. See Example 6.6.3, eg. **S4-T7**.

***Tutor task dialogue-move*** The tutor dialogue-move produced before has a consequence on the choice of the next one. For example, if the previous move is *check-origin-problem*, this will not be produced again in the next turn. See Example 6.6.1, **S4 - T6**.

#### 6.4.1.3 Motivation and Cognitive Load

It consists of GMCL and LMCL. The use of them and their subfields is already explained in Chapter 5. The influence the subfields have on the behaviour of the procedure relates to the GMCL and LMCL as a whole. They are, namely, used for drawing conclusions about the performance of the student, and hence have an impact on the strategy manager feedback with regard to motivation levels and cognitive load considerations. See Example 6.6.1[3].

#### 6.4.1.4 Tutorial Goal Status

**Student task dialogue-move**

- *Time-out* It causes the `diagnostics` subdialogue to be called. We choose to interrupt students after some time, to prevent them from getting lost into a means-ends way of reasoning, which imposes extra cognitive load [Cooper and Sweller, 1987]. The choice of doing a `diagnostics` subdialogue after *time-out* was further suggested by the human tutor of our Wizard-of-Oz experiments. Then, if the student is really in need of help, we can try to hint based on the result of the `diagnostics` subdialogue and point the student to a strategic, schema-promoting method [Cooper and Sweller, 1987].
- *Request assistance* It causes `request-assistance` to be called, which decides whether to treat the request or not, and how (See Section 6.5.5).
- *Request evaluation* It encourages students who ask explicitly for evaluation, which might be a sign that they are feeling demotivated. It informs them how they did in the last domain contribution by doing *signal-domConCat-evaluation*. In Example 6.1, the student asks for a general evaluation of her performance and there's no *domain-contribution* in the same turn, so `Menon` produces a general encouragement.

  (6.1) ...
  **S:** (`request-evaluation`) How am I doing?
  **T:** (`encourage-red-eval`) It's not easy, but you'll get there!

- *Resign* It produces the following behaviour:

  1. If the student input includes some dialogue-move other than *resign*, then we just do *encourage* to treat the *resign* move. Otherwise we concentrate on the other move and ignore the *resign*. Students out of habit or frustration tend to state that they do not know what to do, but do in fact provide some further attempt, which we can make use of to decide on the next feedback.

[3] In the examples, we use the following notation: $LH$ (local-hint count), $LW$ (local wrong-answer count), $LC$ (local correct count)

2. If there is no other dialogue-move in the same turn, first we encourage the student. Then, we do the following:
   (a) If the student performance is low, we prompt the student to try to answer and re-produce the previous hint. The dialogue move *prompt* realised as "Try again" is used to this end. That is an attempt to prohibit students from just asking until they eventually get the answer, and make them think actively and harder.
   (b) For the student who does not make a habit out of resigning (*resign* count $< 2$) or exhibits a good performance (above average, GMCLA $\leq 0.5$), the subdialogue ***diagnostics*** is called to further investigate the reason that the student is not able to progress in the task.
   (c) If the student performance is high but there have been at least two *resign* moves already, we do a ***spell-out-task***. The tutor should not give the answer away to discourage constant resigning, but just another hint is not appropriate either. ***Spell-out-task*** recognises the student's need for help in a more methodical way.

The fields that we looked at so far are employed in the choice of the right pedagogically-motivated dialogue-move and in the choice of the referential perspective (pragmatic vs. conceptual) and substrategy. Let us now turn to the HSS fields that concern domain knowledge and are hence used by the ***performable step***, ***meta-reasoning*** and ***pragmatic*** subtasks. They inform the choice for the INFERENTIAL ROLE and DOMAIN KNOWLEDGE dimensions.

See Example 6.6.1, **S3 - T4**.

#### Previous student task dialogue-move

**Domain contribution** In general, student dialogue-moves except *domain-contributions* are treated differently by the procedure. Domain contribution together with its domain-contribution category are used for informing students of their performance in the particular *domain-contribution* via a *signal-domConCat-evaluation*. This is necessary for making students aware of their progress and not giving them the impression that their effort is pointless. Therefore, we do it after every *domain-contribution*. Moreover, we normally provide *signal-domConCat-evaluation* in combination with feedback that gives students a suggestion on how to deal with errors or inability to move on to avoid demotivating them.

We consider the following domain-contribution categories:

- Correct causes the procedure to either simply accept the answer or to call the subtask ***aligning*** if the student performance is below average (GMCLA $> 0.5$) and after five hints on the same step have been produced, since the last time we called the ***aligning*** subtask. This is the way the system checks whether students really understand the reasoning, when they give a correct answer. The number of hints indicates that the student has just started following, in which case *aligning* reinforces the schema by re-eliciting the reasoning that got us to this point in the task. Another possibility is that the correct answer was a coincidence, so we want to make sure that the student understands why this answer is the correct one, based on the reasoning we have applied. The process of aligning is common practice in the BE&E corpus [Tsovaltzi, 2001]. See Example 6.6.2.

- Partial-answer causes the procedure to do explicit *encourage* if the student performance is lower than average (GMCLA $\leq 0.5$). There are two issues involved here. First, motivation should be taken care of before any content guidance in order to engage the student in the task at all [Keller, 1987; Wood and Middleton, 1975]. Second, empirical findings [Beal and Lee, 2005] suggest that similar behaviour has a negative effect on high performing students, as it inflicts a worry that they will not be able to sustain the hard effort. In general, realisation of *encourage* like positive feedback, "You are doing fine" and effort-sustaining feedback "It is a difficult task, just keep on trying" should be used interchangeably to cater for both the need of the student for appraisal (positive experience) and for the recognition that effort is required, so that the student has the right idea about the correlation between effort and success. (cf. Example 6.6.2, **S1-T2**).

  *Encourage* is hence a kind of a place holder for different methods of motivating the student, which are not explored in this thesis further. From an engineering point of view, though, *encourage* makes their later integration possible. In addition to *encourage*, after a partial answer the procedure chooses a class subtask for hinting.

- Complete-inaccurate effects the production of the hint `discrepancy` if the student's performance is not too bad. It tries to make students realise the discrepancy on their own. If this does not work or if the student's performance is too low, we do an explicit *encourage* and treat complete-inaccurate as wrong. Example 6.2 is taken from participant *socratic1*, and the tutor did do `elicit-substitution` in **T4**. We keep turns **S3**, **T4** and **S4** as they occur in the actual corpus, and only add here *encourage* and *initiate-subtask-subst* that `Menon` currently produces. The proof handled is $A \cap B \in P((A \cup C) \cap (B \cup C))$. The student gives a complete-inaccurate answer, in that she uses the correct Rule of Inference, and all parts of Substitution are present, but they are not correctly substituted. Because she has not been doing well until this turn, `discrepancy` would probably be confusing, so she is asked to apply the rule correctly in **T4**. This hint helps her to correct her mistake and get the step right in **S4**.

(6.2) ...

**S3:** (`complete-inaccurate`) Distributivität von Vereinigung über Durchschnitt: *[Distributivity of union over intersection:]* $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

**T4:** (`encourage`) It's a bit difficult, right? (`signal-ci`) That's not exactly right. (`initiate-subtask-subst`) Let's look at it. (`elic-subst`) Wie hätten Sie diese Regel anwenden müssen? *[How should you apply the rule?]*

**S4:** $(A \cup C) \cap (B \cup C) = (A \cap B) \cup C$

- Wrong initiates a search for the appropriate conceptual or pragmatic subtask. Wrong is also a major factor for calling the *`spell-out-task`*. See Section 6.5.9 and Example 6.6.1, **S1 - T2** and **S5 - T6**.

- Unknown results in calling the *`diagnostics`* subdialogue, to find out what the problem is. However, if the previous move is either *time-out*, or unknown, meaning that *`diagnostics`* has already been called, then we just treat the answer as wrong and call the function CONCEPTUAL. In other words, in these two cases we do not expect any more that students can provide useful information on why they cannot proceed, and we do not want to intimidate them by asking too many questions that they cannot answer.

  In the Wizard-of-Oz experiment, unknown was also followed by a *check-origin-problem*, which is the first move of the *`diagnostics`* subdialogue. See Example 6.6.1, **S4 - T5**.

- Misconception should be treated whenever it is encountered, whether relevant to the current task or not, as a misconception undermines the student's understanding of the domain as a whole. Misconception is treated by the subtask *`explain-misconception`*. See Section 6.5 and Example 6.6.3, **S4 - T7**.

- Missing Basic Knowledge yields the `give-away-basic-knowledge` hint, which provides the missing knowledge (cf. Chapter 4). Missing basic knowledge relates to too basic a concept to be ignored. This is motivated by the fact that the student should not sidetrack too much from the focus of the session for reasons of relevance [Keller, 1987] (cf. Chapter 2), and from research that supports teaching some declarative knowledge when it is necessary for the task [Sun *et al.*, 2001]. In Example 6.3, the proof is $K((A \cup B) \cap (C \cup D)) = (K(A) \cap K(B)) \cup (K(C) \cap K(D))$. In **S2**, participant *socratic5* wanted to apply the rule of inference De Morgan 2 (defined in the lesson material as $K(A \cap B) = K(A) \cup K(B)$), which is the correct rule of inference here, but wrote the definition of De Morgan 1 (defined in the lesson material as $K(A \cup B) = K(A) \cap K(B)$). `Menon` just gives the Basic Knowledge away.

(6.3) ...

**S2:** (`miss-basic-know`) $K((A \cup B) \cap (C \cup D)) = K(A \cup B) \cap K(C \cup D)$

**T3:** (`signal-miss-basic-know`) There's a problem here. (`initiate-task-inf-rule`) You see (`give-away-basic-know`) the De Morgan 2 is the following: $K(A \cap B) = K(A) \cup \mathbf{K}(B)$ (`prompt-action`) Try to go on!

- Step-size is a reason to ask students to justify what they have done. Alternatively, we do the following

  1. If the students' performance is low (GMCLA $\geq 0.3$), or if it is the first step-size in the session, the procedure asks the students if they think they have provided all justification needed, to make sure that students realise what they are doing. To do that, we produce the `discrepancy` realised as a "Why?" question, when the students' performance is above average.
  2. If a step-size has occurred before in the same session or the performance is bad, we check what students think they have done via a *check-origin-problem*.
  3. If the students cannot explain, hence demonstrate a lack of awareness of what needs to be proven, we treat this as a misconception.

  See Example 6.6.1, **S5 - T6** and 6.6.3, **S4 - T7**.

- Irrelevant gives rise to informing the student that the mathematical statement is correct, but it is not useful at this point. That is, we do a *signal-irrelevant* dialogue-move. This avoids confusing students by telling them that something is wrong when it is in principle correct. Apart from that, is treated as a wrong answer.

  When the student gives an irrelevant proof step that we cannot accommodate as a valid step, then it is worth checking if the student is starting a new proof by doing a ***`diagnostics`*** subdialogue. See Section 6.5.1 and Example 6.6.1, **S2 - T3**.

- Ill formed can be treated in two ways.

  1. When the student performance is good (GMCLA $\leq 0.3$), it causes the production of the hint `elicit-ill-formed`, and then `give-away-ill-formed`, which corrects the performable-step formulation.
  2. If the students' performance is much lower than average (GMCLA $>$ 0.3), we just prompt for the next step, ignoring the ill formed so as not to overload students with issues that they cannot handle.

  This behaviour was informed by research in cognitive science [Sweller, 1989; Sweller and Cooper, 1985], arguing against splitting the student's attention. When students are showing a high performance, other aspects of the task can be pointed out that would normally require part of their attention and impose cognitive load. The student, for instance, may have mastered the domain rules and, thus, may not need much memory space for processing them. Therefore, the student can probably handle the cognitive load of correcting the syntax error, which might sidetrack a low-performing student.

In Example 6.4, the proof is $A \cap B \in P((A \cup B) \cap (B \cup C))$. The student writes the first step almost correctly, but misses a bracket, as participant *didactic15* did. `Menon` just ignores this, since we are at the beginning of the session and there hasn't been enough opportunity to assess the level of the student, so we assume a bad model, so as not to discourage the student from the start. As a matter of fact, the tutor in the experiment decided to delve into this mistake, which caused a lot of confusion to the student. `Menon` just prompts the student for the next step.

(6.4) ...

**S1:** (`ill formed`) $P((A \cup B) \cap (B \cup C)) = PC \cup (A \cap B)$

**T2:** (`signal-ill-formed`) That's fine. (`prompt-step`) What's the next step?

- Wrong Linguistic Term yields a `correct-terminology` hint, when the student performance is very good (GMCLA $\leq 0.3$), which is always followed by the appropriate conceptual hint. The hint `correctterminology` repeats exactly what the student said, but for a substitution of the right term for the wrong term that the student has used. If the student performance is poor(GMCLA $\geq 0.3$), we just ignore the wrong linguistic term and treat the domain contribution as correct. Reduction of cognitive load is considered in the same way as it is for ill formed.

  In Example 6.5, the proof is $A \cap B \in P((A \cup B) \cap (B \cup C))$. The student knows the relevant concept (intersection), and the tutor tries to elicit the Subordinate Concept (union) (**T4**). The student provides it, but uses the wrong terminology (**S4**). The tutor in **T5** points out the correct terminology and goes on to the logical next hint.

(6.5) ...

**T4:** (`elicit-sub-con-meta-reas`) Think of what is connected to the intersection and you can use it to prove what you want.

**S4:** (`wrong-ling-term`) The combination of two sets.

**T5:** (`encourage`) It's a bit difficult, (`signal-wrong-ling-term`) but you're right. (`initiate-subtask-sub-con-meta-reas`) OK. (`initiate-subtask-infer-rule`) So, (`prompt-step`) just say "union" instead of "combination" to be accurate. (`elicit-inf-rule`) What rule can you use here, then? (`prompt-step`) Go on, then?

- Near-miss causes the pragmatic hint `discrepancy` to be produced, when the students' level is good. `Discrepancy` just prompts the student to reconsider their answer. The idea is that it might have been a mistake out of lack of concentration, but in any case, something that the student is aware of in principle. Otherwise, if the student still does not realise the mistake, then the subtask *`near-miss`* is called (cf. Section 6.5). The hint `discrepancy` was suggested by our human tutor in the Wizard-of-Oz experiments. See Example 6.6.3, **T2 - T4**.

***Instructional Points***

- Relevant Concept used is employed both for eliciting the Subordinate Concept and eliciting the Rule of Inference. In both cases it constitutes prerequisite knowledge, based on the heuristics we use for promoting schema acquisition (cf. Chapter 2). See Example 6.6.2, **T4 - S4**.

- Subordinate Concept used, with the same rationale, is employed for eliciting the Relevant Concept, provided the Subordinate Concept is known, and for eliciting the inference rule, in combination to the Relevant Concept (cf. Chapter 2). See Example 6.6.2, **T5 - S5**.

  There are two major contradictory views that have a bearing on this decision. Lim and Moore [Lim and Moore, 2002] argue for the elimination of cognitive load via providing appropriate stepwise instruction with the aim of promoting expert-schema acquisition. The evidence for what schemata experts use comes from a study on the comparison between novice and expert problem solving [Chi *et al.*, 1982]. Lim and Moore's view argues for always dealing with the Relevant and the Subordinate Concepts, because they are instructional points.

  On the other hand, Motivation Theory [Keller, 1987; de Vicente and Pain, 1998; Weiner, 1992] suggests to use as little instruction as possible in order to enhance the student's feeling of achievement. That is an argument against always tutoring the Subordinate and the Relevant Concept if the student already knows the Rule of Inference to apply. The knowledge has been proceduralised and there is no need to make it declarative again [Lewicki *et al.*, 1992], which would also add unnecessary cognitive load. There is also the danger that the schema that the student has is somewhat different from the one we are trying to elicit. Therefore, we only use the Subordinate and Relevant Concept when it is necessary in order to promote procedural knowledge for choosing the applicable Rule of Inference (see also 2). Thus, we combine the contradictory aspects of cognitive load elimination and increasing motivation.

- Domain Relation used causes the corresponding passive Domain Relation hint to be produced. This is a way of eliciting the domain concepts when a related concept is present in the student input. As a side-effect, it reinforces the awareness of the domain structure. Example 6.6.3, **T2 -**

**T4** shows how a domain relation is used to identify a near-miss *domain-contribution*.

- Rule of Inference used is a prerequisite for eliciting the Substitution for the proof step. See Example 6.6.2, **T6 - S6**.

- Inverse Rule used helps to inform students of the fact that they are using the inverse to the Rule of Inference. The corresponding hint is produced. In Example 6.6, the proof is if $A \subseteq K(B)$, then $B \subseteq K(A)$. In **S2**, which is taken from participant *socratic23*, the tutor noted that the student started from the wrong direction of the implication, and that she should be informed accordingly. Here we show `Menon`'s behaviour, after the incorporation of the tutor's comment. The tutor in **T4** informs the student of having used the inverse rule and we are now inside subtask ***`inf-rule-meta-reas`***, so the hinting will go on at the level of the Rule of Inference.

(6.6) ...

**S2:** (`wrong`) $A \subseteq K(K(A))$

**T3:** (`signal-wrong`) That's not quite right. (`give-away-inv-rule`) The rule that you want to use applies to the opposite direction of what you need here. (`prompt-action`) Try to go on now!

- Technique used is employed to elicit the Rule of Inference. See Example 6.6.2 **T6 - S6**, where the technique is that the if-then has to be eliminated.

- Starting Point used is employed to elicit the Proof Step, as it constitutes part of the meta-reasoning for arriving at it. See Example 6.6.2, **T1 - S1**.

- Premise used indicates an awareness of the basic meta-reasoning of what the proof problem is, and is used in the process of eliciting the performable step as a whole. See Example 6.6.2, **T2 - S2**.

- Conclusion used has the same effect as premise used. Premise and conclusion are used together as one depends on the other. It aims at making the student aware of the problem solving situation before embarking on finding out which rule applies to the situation. See Example 6.6.2, **T2 - S2**.

- Directness used is employed in eliciting the Proof Step, which can be a subproof. It is the most abstract level of a technique, which does not refer to manipulations inside one step. The relevant hint is only used when the proof to be performed is indirect, as direct is the default case, and there is no reason to confuse the student. See Example 6.6.4, **S8 - T9**.

- Direction used, like directness used, is employed in eliciting the Proof Step. It influences the choice of inference rule. See Example 6.6.2, **T3 - S3**.

Premise used, conclusion used, directness used and direction used all trigger hints that refer to the meta-reasoning of the Proof Step. However, unlike the performable-step hints, which are produced last in the hinting process for one step, these hints are produced at the beginning. This is due to the fact that they address the step as a whole and understanding them is necessary for understanding the following instructional points in the proof (cf. Section 6.5.4).

#### *Pragmatic information*

- Different Theory causes `Socratic` to ask the student to take for granted the information that belongs to a different theory [Collins and Stevens, 1982]. The "local axiomatics" notion [Wu, 2001] motivates this, as the student should not be dealing with too many things at the same time. Practically, in the Diagram Configuration model [Koedinger and Anderson, 1990] for geometry proofs, ignoring algebraic inferences resulted in a reduction of the search space, which translates into a reduction of cognitive load. In Example 6.7, the proof is if $A \subseteq K(B)$, then $B \subseteq K(A)$. The student requests an explanation of why in case of implications we can assume the premise. This would require going into logic, so the tutor just indicates that this is nothing to concern the student and urges her to go on.

  (6.7) ...

  **T4:** (`signal-wrong`) That's not quite right. (`give-away-proof-step`) We assume that $A \subseteq K(B)$ and prove that $B \subseteq K(A)$

  **S4:** (`req-ass-dif-theory`) Why are we allowed to do that?

  **T5:** (`initiate-subtask-req-ass`) OK. (`correct-info`) Let's not look into this now. (`prompt-action`) Try to move on now.

- Same Domain Information results in referring the student back to the session. Based on what the values of the same information are and whether the student level is good enough, the tutor points the student to the previous occurrence of that domain information. This is done with the pragmatic information hint `point-backwards`, since the student is only asked to look back and find the information needed for the proof currently. If the student is unable to track it, then the information is given away with a passive conceptual hint. Other pedagogical considerations are also taken into account. For instance, the student is pointed back to a previous occurrence of the same Domain Technique only if the technique was also applied on the same Relevant Concept. This is done to increase the connection between the two uses.

  To demonstrate how Same Domain Information works, in Example 6.8, the proof is if $A \subseteq K(B)$, then $B \subseteq K(A)$. The student knows the Relevant Concept and the Rule of Inference (**S1**) right after the initiation of the task, so the tutor elicits the Substitution (**T2**). The student provides it (**S2**) and with that the first step is completed. In **S10**, the student has to apply the same rule and states the Relevant Concept (if-then) and the Subordinate Concept (the expression that has to be proved). So, the Rule of Inference is elicited (**T11**), but the student names the wrong Rule of Inference (**S11**). Since the Rule of Inference here constitutes the Same Domain Information, the hint `point-backwards` is produces, as described above.

| | |
|---|---|
| | ... |
| **S1:** | (`partial-answer`) I have to use if-then and the rule let x and prove y. |
| **T2:** | (`encourage`) Great! (`signal-pa`) you're on the right track. (`initiate-subtask-subst`) Let's look at it. (`elicit-subst`) Try now to apply the rule (rule of inference) to the expression. |
| **S2:** | (`correct`) Assume A is a subset K(B), and show B is a subset of K(A)! |
| **T3:** | (`signal-accept`) Correct! (`prompt-step`) What's the next step? |
| | ... |
| **S10:** | (`partial-answer`) Now I have the if-then again, and I have to think of what I need to prove. |
| **T11:** | (`encourage`) Great! (`signal-pa`) you're on the right track. (`initiate-subtask-inf-rule`) Let's see. (`elic-inf-rule`) What rule can you use here, then? |
| **S11:** | (`wrong`) The de Morgen. |
| **T12:** | (`signal-wrong`) That's not quite right. (`point-backwards`) We have already looked at a similar case in this proof. Have a look back and try to move on with the proof. (`prompt-action`) Give it a go. |
| | ... |

(6.8)

- Ordered List Available causes the pragmatic hint `orderedlist` to be produced, which makes use of the pragmatic information available. If the student does not respond correctly to this hint, the pragmatic hint `refer-to-lesson` is produced, as this information is in the lesson preparation material. Students can learn to use this material better when they know what they are supposed to be looking for. In Example 6.9, we are assuming that the student has to do an inductive proof, and is being asked about the first case in **T3**, but cannot provide it. The tutor in **T4** tells her that there are three cases and asks her for the first in order. When the student provides a wrong answer again in **S4**, the tutor resorts to referring her to the specific place in the lesson, which she has to read.

(6.9) ...

**T3:** (`elicit-inf-rule`) Which rule can you apply, then?
**S3:** (`resign`) I don't know.
**T4:** (`encourage`) It's a bit difficult, right? (`ordered-list`) You need to analyse the problem into three cases. Which is the first one?
**S4:** (`wrong`) Where I talk about all instances.
**T5:** (`encourage`) Don't worry. (`signal-wrong`) That's not quite right. (`refer-to-lesson`) Just, go back and read about the cases of an inductive proof again. (`prompt-action`) Give it a go.
...

- Unordered List Available causes the pragmatic hint `unordered-list` to be produced if the student's *domain-contribution* is a partial answer with some parts missing. It aims at allowing students to figure out the rest of the elements of the unordered list by themselves. If after that the student does not answer correctly and when the student level is high (above average), the hint `narrow-down-choices` is produced. This gives students the chance to provide the correct answer by practically reminding them of the options and narrowing the search space for the potentially correct answer.

  In Example 6.10, the proof is $A \cap B \in P((A \cup B) \cap (B \cup C))$. The student in **S3** asks for the Rule of Inference, after having been given the Relevant Concept. The tutor first informs her in **T4** that there are three attributes (as there were in our lesson material in the Wizard-of-Oz experiment), hoping to remind her of them and prompt her to name them. When the student in **S4** does not provide the right answer, the tutor gives her a choice between two of the attributes.

(6.10) ...

**T3:** (`give-away-rel-con`) You have to use powerset.
**S3:** (`req-ass-inf-rule`) What exactly about the powerset do I have to use?
**T4:** (`unordered-list`) There are eleven attributes, which one do you need?
**S4:** (`resign`) I don't know.
**T5:** (`narrow-down-choices`) Do you need the definition or the attribute that says that if $A \subseteq B$, then $P(A) \subseteq P(B)$?
...

### *Proof Status*

- Proof Step Completed causes the HSS to be reset, if the student performance is high. The new proof step will start with new information on the student performance for that step. For a low student performance, the subtask *aligning* (cf. Section 6.5.8) is called for the completed step. See Example 6.6.3, **T - last**.

- Proof Completed is used to do a recapitulation of the proof for students with a bad overall performance, and to reset the HSS. Apart from assigning the student with the right level of task, whose choice we assume, we give students the opportunity to start without any preconceptions on how they will perform on the task. See Example 6.6.1, **S14 - T17**.

## 6.5 Substrategies

We first give an overview of the notion of a substrategy. Then we describe the different substrategies and provide motivation for their behaviour according to our teaching strategy.

The following substrategies are implemented:

**Subdialogues:**

1. *Diagnostics*, which attempts to identify the problem that causes the student's problematic behaviour

**Subtasks:**

1. *Pragmatic*, which picks a hint based on any pragmatic information available
2. *Performable-step*, which picks an appropriate hint to address the performable step
3. *Meta-reasoning*, which picks an appropriate hint to address the meta-reasoning of the step
4. *Request-assistance*, which deals with requests for assistance from the student
5. *Near-miss*, which handles near-miss domain contributions
6. *Explain-misconception*, which handles diagnosed misconceptions
7. *Aligning*, which tries to make sure that the student has understood a previous series of hints
8. *Spell-out-task*, which guides the student closely through completing the proof step
9. *Recapitulation*, which recapitulates completed steps and the proof, at the end of one session

Substrategies are selected based on HSS situations, but also depending on the student performance by means of the GMCLA. Performable-step subtasks are the core of the teaching strategy and assist schema acquisition. Note that every substrategy can only provide the hints that are relevant to its function. For example, *meta-reasoning* will not produce any performable-step hints, as they cohabit the same hint dimension and, hence, are counter exclusive.

### 6.5.1 *Diagnostics* Subdialogue

**Hinting Situations** *Diagnostics* is the only subdialogue currently initiated by the tutorial manager. It is called when the system cannot categorise the student's input with the precision necessary for deciding upon appropriate feedback.

The cases it handles are the following:

1. when there is an unknown *domain-contribution*
2. when there is a *time-out*
3. when students resign, so we cannot know what the specific problem they had with the task was
4. when students perform a step-size (does not prove some in between step), in which case we do not know if they just consider the step trivial or if they do not realise that normally they would need to prove it
5. when the step is irrelevant but correct, in which case we check the possibility that students are attempting a new proof
6. when students request assistance and their performance is low, in which case we try to find out what the actual problem is
7. when the *domain-contribution* is wrong, after a point-to-lesson hint, because the problem might not be what we thought, and this might be the reason why the point-to-lesson did not work

**Behaviour** *Diagnostics* starts with a simple *check-origin-problem* dialogue-move. This gives students the opportunity to explain the problem they are having or even show that there is no problem. However, if there indeed appears to be a problem after the *check-origin-problem* move, but it is still unclear exactly what it is, the appropriate active meta-reasoning eliciting hints are called (based on the domain information being handled). As soon as a student inputs a *domain-contribution* that can be evaluated, we start providing the active meta-reasoning hints. However, the aim of those hints is to find out which domain information (Instructional Point) the student has a problem with. When the procedure hits on a hint that the student cannot answer, the respective subtask is called to deal with it. By eliciting such information, we avoid asking the student "What is wrong" type of questions, which are normally ineffective, as students are usually bad at evaluating their own problem solving [Aleven and Koedinger, 2000b]. Moreover, a most common phenomenon is that students state that they do not know the answer, presumably hoping that they will get it from the tutor with minimal effort, instead of trying to find it. It is telling that in the DIALOG corpus eight out of nine "Do you know..." questions were answered with "no" by the students[4]. Therefore, it is the tutor's responsibility to find out what the problem is, as well as to try to deal with it. The strategy also skips the original *check-origin-problem* in cases where ***diagnostics*** is called after `point-to-lesson`, or after *request-assistance*. In both instances the student level is too low to pose the abstract question "What is the problem?". In fact, in the case of a *request-assistance*, students think they know what the problem is, but we distrust their judgement.

Although the NL realisation of hints is not the subject matter of this thesis, let us note that both the dialogue-move *check-origin-problem* and the meta-reasoning hints may be realised as "Wh-questions". The *check-origin-problem* dialogue-move is produced with a parameter, namely the student input that does not allow a categorisation for appropriate feedback and causes the subdialogue to be called. For example, the most obvious case is when the student input is not understood at all at the domain knowledge level (as opposed to the linguistic level), in which case we want to ask students to explain what they meant, rather than what problem they are having.

**Justification** The idea of the diagnostics subdialogue is to understand the exact problem of the student so as to be able to decide on the feedback. Therefore, it facilitates the choice of subtasks. The dialogue manager alone cannot handle the subdialogue, as pedagogical knowledge is necessary for it.

**Examples** See Example 6.6.1, **S4 - T6**.

### 6.5.2 *Pragmatic* Subtask

**Hinting Situations** Pragmatic hints in general motivate the student, as they are good for providing minimal feedback based on the student input. Therefore, ***pragmatic*** is called whenever possible, that is when some pragmatic information is available (cf. Chapter 5).

**Behaviour** The subtask ***pragmatic*** picks a hint based on the pragmatic information available.

1. Wrong linguistic term gives rise to the hint `correct-term`, which gives the term away.

2. A complete-inaccurate *domain-contribution* produces a `discrepancy`, for high level students.

[4] In the case students knew the answer, they provided it.

3. An ordered list available results in the corresponding hint, and if this is not answered in a `refer-to-lesson`, so that the student can benefit by consulting the study material that includes the list (cf. Chapter 5).

4. An unordered list available causes the production first of the corresponding hint, and then of `narrow-down-choices`, to help the student find the missing element of the list.

**Justification** A pragmatic hint might help the student get back on track with minimal possible feedback (cf. Chapter 4). Pragmatic hints are often considered as prompts and are favoured by tutors [Tsovaltzi, 2001; Graesser *et al.*, 2005].

**Examples** See Example 6.5.

### 6.5.3 *Performable-step* Subtask

**Hinting Situations** *Performable-step* is a subtask that comprises four class subtasks, namely *domain-object*, *inference-rule*, *substitution* and *proof-step*, and the main function controlling the choice of class subtask called PERFORMABLE-STEP. Performable-step class subtasks correspond to the main instructional points, apart from Domain Relation. There is no class subtask for Domain Relation for the performable step, as Domain Relations only have a function related to meta-reasoning in our teaching strategy.

In function PERFORMABLE-STEP, instructional points are addressed in an order from more abstract to closer to the performance of the step, as the more abstract information consists of the heuristics for deriving the performed step. For example, the *domain-object* class subtask, which is an instructional point used for finding the Rule of Inference, is called before the *inference-rule* itself. We consider some information known:

1. if the hint that gives it away has been used

2. if the student has used the information

3. if it is a prerequisite for some other piece of information, which is known

**Behaviour** The *performable-step* class subtasks choose a hint for production from the ones available for the main instructional point, which they deal with. Hints are chosen based on logical priority and elicitation status. The former makes sure that the information offered by the hint is not given away before any prerequisite information, given by another hint. This translates into, for instance, only eliciting explicitly the Rule of Inference, when the Relevant and the Subordinate Concepts are already known (cf. Appendix E).

In addition, we elicit when the student performance is good enough to allow inferring a high level of motivation and the possibility that the student is able to answer the eliciting hint. If not, we directly provide the passive hint, which just gives some information to try to help the student derive the proof step.

Another aspect of the ***performable-step subtask*** is that it calls the ***meta-reasoning*** subtask at appropriate places for every instructional point (cf. Section 6.5.4).

**Justification** The general philosophy of the ***performable-step*** subtask is captured in the heuristic blueprint presented in Chapter 2. Elicitation status refers to the fact that an active hint is always produced before the equivalent passive one. With regard to this, observations from the BE&E show that if the concept has already been dealt with previously in the tutorial session [Tsovaltzi, 2001], established (or grounded) then the tutor will give it away. If not, then the tutor will elicit it [Freedman *et al.*, 1998]. The more elaborate way in which we deal with the elicitation status choice is informed by the research results from the learning sciences. Hence, we take the student performance into account.

**Examples** See Example 6.6.2, **S4 - T8**.

### 6.5.4 *Meta-reasoning* Subtask

**Hinting Situations** ***Meta-reasoning*** is called when the domain content of a specific hint has not been dealt with yet, but the student seems to need extra supportive instructions to arrive at the step. To infer this, the procedure consults the GMCLA and if this is less than average (GMCLA $\leq 0.5$), the ***meta-reasoning*** subtask is entered, by directly calling the subtask that corresponds to the instructional point being handled at that moment.

Otherwise, since students do not seem to need the extra support, we choose not to interfere with that level of cognition (we provide little instruction), to allow them to build their own schema on the most minimal feedback possible.

**Behaviour** Inside the ***meta-reasoning*** subtasks, we elicit the meta-reasoning only if the student performance is high (GMCLA $\geq 0.75$), since eliciting the meta-reasoning may impose cognitive load that cannot be handled by students of a poorer performance. Otherwise, we give it directly away.

The ***meta-reasoning*** class subtasks are called via the corresponding class subtasks of the ***performable-step*** subtask, because they reinforce the main instructional points of the ***performable-step*** subtask. For the same reason, the meta-reasoning for a class subtask covers also the content of performable-step hints that belong to the same class subtask. Meta-reasoning hints also comply with the subordination, hence the ***meta-reasoning*** subtask chooses a hint based on this relation. So, the hint `subordinate-concept-meta-reasoning` is produced before the hint `inference-rule-meta-reasoning`.

The ***proof-step meta-reasoning*** addresses the step as a whole. It provides an overview of the proving procedure and is produced at the beginning, as an exception, in order to motivate the whole step. This means that the hinting for a step starts with the ***proof-step meta-reasoning*** hints and ends with the ***performable-step*** hints of the same class. This reflects the proving cycle of looking forward at what has to be proved, and back again at what needs to be done locally to reach that goal [Wu, 2001].

Finally, the meta-reasoning for the step comprises passive meta-reasoning hints from all classes that are general enough to be used in the derivation of every step. `Proof-step` hints, in particular, are only produced if any of them have been produced in the course of tutoring the Proof Step instructional point. If not, we do not trouble students with that abstract level, which they were already comfortable with.

**Justification** Giving meta-reasoning hints is motivated by cognitive load theory. More advanced learners may be hindered in the process of learning by the same information which helps beginners. For the advanced learners this information will require unnecessary processing load for something that has already been automated via a schema [Paas *et al.*, 2004]. In addition, meta-reasoning plays the role of abstract instructions increases transferability, as they help the promotion of a more general schema [Robertson, 2000].

The need to include the *`meta-reasoning`* was explicitly recognised by the human tutor in our Wizard-of-Oz experiments (cf. Section 1.4.2). Moreover, additional support for the *`meta-reasoning`* comes from the mathematics research. Wu [Wu, 2001; 1996] favours making the students aware of the logical reasoning behind proving.

This two-level approach of providing both performable-step and meta-reasoning hints complies with the synergistic view of bottom-up and top-down learning [Sun *et al.*, 2001]. The *`performable-step`* subtask may help the more abstract meta-reasoning to be built bottom-up. If not, *`meta-reasoning`* points to the concrete performable step top-down (cf. Chapter 2).

**Examples** See Example 6.6.1, **T2 - T7**.

### 6.5.5 *`Request-assistance`* Subtask

**Hinting Situations** The subtask is called after the student task dialogue-move *request-assistance* in order to decide how to deal with the request. The initiative of the student to request some specific assistance takes us out of the conceptual method of deriving a proof.

**Behaviour** The kinds of assistance that we provide when explicitly requested are all possible values of the HSS field *assistance-requested* (cf. Chapter 5).

1. If the student performance is lower than average, and there has been at most one previous *request-assistance* so far, we give the answer away, in the form of a `correct-information` pragmatic hint, since this does not really follow the main conceptual schema building and students should stop troubling themselves with it. The student has not tried to take advantage of the fact that we do give answers away and would be demotivated if we did not give the answer.

2. If the student performance is good, we encourage the student, who feels in need of assistance, and we call the ***diagnostics*** subtask, to find out the exact problem before we can help the student.

3. If the answer to the student's inquiry belongs to a different theory than the one being taught, then the system informs the student that the inquiry will not be dealt with via a `correct-information` and, thus, asks the student to take it for granted.

4. If what the student asks is not relevant to the current session, that is it is neither one of the domain information captured in the HSS (an Instructional Point) nor some Basic Knowledge, the system tells the student to concentrated on the current goal (ignoring the question), and re-states the last tutor-task DM, which the student is supposed to respond to or produces a *prompt* to carry on with the task.

5. If what the student has asked about is some piece of domain knowledge that has been dealt with before in the course of this proof step, we tell the student to look back for it with the pragmatic hint `point-backwards`, in order to enhance the student's ability to bring all the information of a session together and take this piece of information for granted.

6. When the student inquires about some basic knowledge or something ill formed dealt with before, we just give the information to the student by providing a pragmatic hint `correct-information`, as the student should now accept the information, and move on with the task. The `correct-information` hint reproduces the hint that gave this information away before.

7. If students after the `point-backwards` still do not know the answer to their *request-assistance*, the system gives the answer away in the same fashion. For example, if the Subordinate Concept was first mentioned by using the hint `give-away-relevant-concept-meta-reasoning`, we reproduce that, instead of this hint instead of `give-away-subordinate-concept`.

**Justification** `Point-backwards` lets students note that they should now accept the information that has appeared before and move on with the task, taking this piece of information, which has been dealt with before, for granted. This realises the inferential role function of the hint `point-backwards`. *Prompt* and `point-backwards` are used in the same context as motivation fortifiers in the model of Beal and colleagues [Beal and Lee, 2005], which deals exclusively with when to provide help or not, taking into account motivational aspects of the student's behaviour.

We do not point backwards in the case of basic knowledge or something ill-formed dealt with before (represented in the previous used hint and its specifications). This might be confusing. Inquiring about some Basic Knowledge through a *request-assistance* dialogue-move means that students are generally not following and are not performing well. Informing them that we have talked about what they ask again would probably demotivate them. Their level is not good enough for such fine-grained tutoring. It is also difficult to refer clearly to a previous occurrence of such issues.

When giving the answer away after a fruitless `point-backwards`, the student can easily spot the information in question and make the connection without being confused. Note that the discourse module could add here something along the lines of "Remember, we said that..." in the realisation of `correct-information` produced after a `point-backwards`, based on other discourse issues to be taken care of and the dialogue model for the genre[5].

**Examples** See Example 6.6.3, **T7 - T9** and Example 6.6.4, **S9 - T10**.

### 6.5.6 *Near-miss* Subtask

**Hinting Situations** *Near-miss* is a subtask, which causes a slight sidetracking from the main reasoning method followed towards the completion of the proof step. It potentially occupies multiple turns.

**Behaviour** After a near-miss *domain-contribution*, `discrepancy` is used, which is a pragmatic hint, to point out that there is some discrepancy in the almost correct answer.

1. If the student does not realise the mistake with the use of the `discrepancy` hint, then the next hint depends on the type of the near-miss answer. The *near-miss* chooses the appropriate hint, which tries to help the student more to identify the error:
   (a) If the near-miss corresponds to a domain relation, the equivalent passive meta-reasoning domain-relation hint is produced, to inform the student which the mistake is.
   (b) If the near-miss refers to a wrong linguistic-term or an ill-formed structure, the student is informed of the corresponding mistake, but apart from that the next appropriate hint is produced to help the student proceed with the proof as normal. For students who can still correct the near-miss, this information gives them the opportunity to do it. Otherwise, providing the next hint is a way of not paying further attention to the minor mistake of the near-miss.
2. If the student's response to the near-miss hint is still not correct, we return to the *proof-step* subtask.

[5]This might be inappropriate, for instance, if the student has previously answered the `point-backwards` with "I don't remember". In that case it would be frustrating to ask students if they remember just after they have stated that they do not (even though it is a rhetorical question). This is the kind of flexibility our hinting modelling provides. A fine distinction, which might have a lot of cognitive impact [Moore, 1993].

**Justification** The `near-miss` subtask serves the relevance and confidence aspects of the student's motivation [Keller, 1987] (cf. Chapter 2). However, there is no obvious pedagogical value on insisting that near-misses are resolved, as this would be counter-productive if the student does not understand the mistake easily.

**Examples** See Example 6.6.3, **T2 - T4**.

### 6.5.7 *Explain-misconception* Subtask

**Hinting Situations** The *explain-misconception* is called when a misconception occurs, depending on the GMCLA.

**Behaviour** The subtask keeps checking for misconceptions by doing *check-origin-problem* for the original input first and after that for the following student input, until no misconception is input. Otherwise, the *explain-misconception* explains each misconception that is encountered.

At the moment we automatically recognise the following misconceptions (cf. Chapter 5):

1. a near-miss connected to any of the Domain Relations that we define
2. a hypotaxis *domain-contribution*
3. a primitive *domain-contribution*
4. a step-size *domain-contribution*
5. an inverse-rule *domain-contribution*

We originally treat those student responses as plain mistakes. The student might get momentarily confused and correct it after the hint that draws attention to it. If, however, this does not happen, then we check more explicitly whether the mistake is really a misconception by calling subtask *explain-misconception* on it.

Apart from the above misconceptions, the subtask can also treat any other misconceptions that may be recognised, for instance, based on a database of domain-specific misconceptions [Stevens *et al.*, 1982; Burton, 1982].

**Justification** The idea behind checking for consecutive misconceptions is that one misconception is commonly based on another one, which we want to discover with this opportunity [Stevens *et al.*, 1982]. The first *check-origin-problem* also gives students the opportunity to correct themselves, in case the misconception was wrongly assessed.

**Examples** See Example 6.6.3, **S4 - T7**.

### 6.5.8 *Aligning* Subtask

**Hinting Situations** The *aligning* subtask is called, provided that the student performance is low, when:

1. The student gives a correct answer for the first time after at least five hints.
2. A proof step is completed by the student on which some hints have been produced.

These situations give reason to believe that maybe the student does not fully understand the followed process for arriving at the answer.

**Behaviour** *Aligning* subtask goes back as many hints as there have been produced for the step until now, provided there are at least five hints on which no aligning has been done yet, and reproduces their equivalent active hint. If the student answer is anything but correct, it gives the answer to that hint away and produces the next active hint. The following issues are central to the subtask:

1. *Aligning* is realised by the dialogue-move *align* the active hint already produced in the session when eliciting the answer, or the equivalent passive hint when giving away the answer.
2. Not all previous hints are addressed. Only one of the hints that address the same point is produced in the course of *aligning*. This is no more as detailed as the subtasks that originally produced the hints, namely *performable-step* and *meta-reasoning*.
3. The meta-reasoning hint is preferred if it happens to be available for reproduction to the performable step equivalent, as the former includes the latter.
4. We also do not reproduce pragmatic hints, passive domain-relation hints, or any of the hints produced in the subtasks that are not directly related to the line of reasoning, as the aim is to concentrate on the reasoning until now and provide a run-through for it.

At the moment, we are focusing on the repetition of the minimal number of hints to avoid overload and frustration. If it proves insufficient we will consider the addition of all-proof-step-meta-reasoning for a rounder schema acquisition, even when such hints have not been produced during the session before.

**Justification** *Aligning* is a common tactic used among tutors who prefer dialectic tutoring. The idea is that if the student was able to come to grips with the last hint and provides the answer, she might have started to follow the correct reasoning and be able to understand the previous steps better.

Gertner and colleagues [Gertner *et al.*, 1998] use the notion of making sure that the student has learned. In their probabilistic student model, they use different weights for representing that the student has acquired some piece of knowledge when the student makes use of it directly, as opposed to that piece of knowledge being given away by a hint. In the latter case they use a strategy similar to `aligning`.

**Examples** See Example 6.6.3, **T11 - T17**.

### 6.5.9 `Spell-out-task` Subtask

**Hinting Situations** This subtask is called when the student shows some unwillingness to proceed, or the GMCLA allows us to infer that the student is frustrated and needs closer guidance. We come out of the `spell-out-task` subtask if the number of correct answers for one step exceeds the number of wrong answers, or if the proof step is completed.

**Behaviour** `Spell-out-task` consists of taking the student through all the substeps of the task (all instructional points that need to be addressed) and giving the answer away more easily. If there is a wrong answer, it gives the corresponding sub-answer away. On the whole, it treats all *domain-contributions* as wrong, apart from the categories, ill formed, wrong linguistic term, misconception and missing basic know. That makes the substrategy less eliciting and more guiding.

In case of student input other than *domain-contributions*, the behaviour is as follows:

1. If students resign from the task, the system asks them to read the lesson material again, before the close guidance can carry on.
2. If there is a *request-assistance*, there are two cases:
   (a) When the answer to the assistance requested has been dealt with before in the session, the subtask provides it again by a *correct-information*.
   (b) Otherwise, the answer is not provided, since `spell-out-task` is going to lead the student to it at the appropriate point, anyway.
3. A *time-out* is treated as a wrong *domain-contribution* and not via `diagnostics`.
4. A *request-evaluation* results in a *signal-evaluation*, after first encouraging the student.

**Justification** The motivation for applying the more guiding subtask (spell-out-task) is that eliciting is still present, but the student is not left unguided at any point. This, of course, means that the student also does not have the task initiative anymore.

`Spell-out-task` replaces the switch to the didactic strategy, which was used in the Wizard-of-Oz experiments, as it is far more consistent with our Socratic teaching model. It carries on with hinting, hence it does not require a huge change in the style of tutoring, and provides a better opportunity to students to start getting back on track with the help of hints, and in the end complete as much of the task by themselves as possible.

**Examples** See Example 6.6.3, **T8 - S10**.

### 6.5.10 *Recapitulation* Subtask

**Hinting Situations** `Recapitulation` is called at the end of every proof step and at the end of the proof.

**Behaviour** It recapitulates the proof step or the proof respectively, based on the particular proof that the student develops, as well as on the guidance that has been provided in terms of hints. We produce all main meta-reasoning hints and all hints that were produced in the course of the proof step as these summarise the reasoning for the step. The recapitulation of the proof should be a summary of the proof performed by the student, which is a matter of NL generation. There are tools that provide NL generation of proofs, e.g. [Fiedler, 2001b; 2001a].

**Justification** Recapitulating the proof is really another reminder of the steps which have been followed during the proof. It is the most standard tactic used by human tutors, even the more didactic oriented ones. The tutor recapitulates in the best possible way, using correct terminology and causal links appropriately. Therefore, `recapitulation` also shows students the best way to formulate proof steps. Wood and Middleton [Wood and Middleton, 1975] refer to this as "demonstration" (p. 98).

In addition, `recapitulation` is a way of increasing the perceived relevance of the task. It helps to increase the continuity of the task and its different steps, thus contributing to the motivation of the student [Keller, 1987]. Delclos and Harrington [Delclos and Harrington, 1991] have experimentally tested recapitulations by using review and discussion as part of their training phase. The relevant condition performed better in subsequent problem solving.

**Examples** See Example 6.6.1, **T - last**.

## 6.6 General Examples

In this section we examine some examples which Menon produces with more steps involved to demonstrate how the strategy develops over multiple turns. Example 6.6.1 deals with tutoring the first step, and Example 6.6.2 the first and second step of the same proof. They depict the behaviour of Menon for different student responses and different student levels. Example 6.6.3 still uses the same proof, to allow comparison, but develops over more steps and shows more explicitly how substrategies work. Finally, Example 6.6.4 explores the different possibilities for students to take back what they did before and the corresponding behaviour of Menon. In some cases there is no corresponding NL formulation for each part of the feedback, which means that the formulations for two parts have been collapsed (e.g., Example 6.6.3, **T8**). For every example, we explore the student and the tutor turn that follows it in pairs. We include comments first for the student turn and then for the tutor feedback to it. We also provide the uncommented version of the example, for a better overview. The exact dialogue moves which constituted Menon's output can be found in brackets proceeding their NL formulation. For brevity, we do not show the specifications of the dialogue moves that Menon also provides and were detailed in Chapter 4. Such specifications are represented in Menon as dialogue-move parameters (:dmpar in Figure 6.2). All NL formulations are just examples that we made up for the purposes of demonstration, as Menon does not provide NL output. The same holds for discourse cues, where they are not annotated with a dialogue act. The annotated ones are provided by Menon. Note that the instructional points addressed in the examples and the order in which they are addressed depict different applications of the instructional blueprint introduced in Section 3.2.

Before we move on to the examples, we list the proof steps in the proof presented here and the rules of inference for them in the form of `give-away-inference-rule` hints, for the orientation of the reader. The steps are given at a very low level of abstraction. Step 1, for instance, might seem obvious, however all students in the Wizard-of-Oz experiment had to be explicitly instructed on it as presented here. The German sentences are actual formulations by the tutor in the experiment.

**Give-away-inference-rule hints for all proof steps**

The task was given in the form:

Wenn $A \subseteq K(B)$, dann $B \subseteq K(A)$

*[If $A \subseteq K(B)$, then $B \subseteq K(A)$]*

1. Zunächst setzen wir die Gültigkeit von $A \subseteq K(B)$ voraus, denn dies ist die Voraussetzung, und wir beweisen, dass $B \subseteq K(A)$ daraus folgt.
   *[To begin with, we assume that $A \subseteq K(B)$ holds, as this is the premise, and we prove that $B \subseteq K(A)$ follows.]*

$$[A \subseteq K(B)]$$

$$\frac{\begin{array}{c}\vdots \\ B \subseteq K(A)\end{array}}{A \subseteq K(B) \Rightarrow B \subseteq K(A)}$$

(give-away-inf-rule) You have to get rid of the implication (if-then relation) by assuming the hypothesis and proving the conclusion from that.

2. Dann müssen wir zeigen, dass alle Elemente aus $B$, auch in $K(A)$ sind.
   *[Then we have to show that all elements of B are also in K(A)]*

   (give-away-inf-rule) Apply the definition of subset.

3. Dann nehmen wir ein beliebiges Element $x \in B$ und zeigen, daß dieses auch in $K(A)$ sein muß.
   *[Then we take an arbitrary element $x \in B$ and show that this must also be in $K(A)$]*

   (give-away-inf-rule) You have to use the rule that tells you how to prove something for all elements.

4. Dann ist $x$ nicht in $K(B)$
   *[Then $x$ is not in K(B)]*

   (give-away-inf-rule) Use the definition of complement.

5. Aber wenn $x$ ist nicht in $K(B)$ und $A \subseteq K(B)$, das heisst, dass $x$ ist nicht in $A$
   *[But if $x$ is not in $K(B)$ and $A \subseteq K(B)$, that means that $x$ is not in $A$]*

   (give-away-inf-rule) Here you have to use that $A \subseteq K(B)$ and you know that $x \in B$.

6. Wenn $x$ aber nicht in $A$ ist, so ist es in $K(A)$.
   *[But if $x$ is not in $A$, then it is in $K(A)$]*

   (give-away-inf-rule) Use the definition of complement

### 6.6.1 First proof step with no correct answers

In this example, the student responses up to **S5**, where the step was given away, are taken from the DIALOG corpus (participant *socratic23*). Apart from **S5**, for which a new category was introduced after the analysis of the corpus, the categorisation of the student input is the one done by the tutor-wizard. **S6** is a wrong answer taken from participant *socratic13*, in order to demonstrate the prolongation of the session by one turn. In all places, the feedback is the one currently produced by `Menon`.

### Commented Example

At the beginning of the session the dialogue is initiated.

**T0:** (`initiate-dialogue`) Hello!
**S0:** Hello!

Since the HSS is empty, the tutoring task is initiated, and the student is prompted for the next step, as the proof is not complete.

**T1:** (`initiate-task, prompt`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset of K(B), then B is a subset of K(A)!
**S1:** (`wrong`) $A \subseteq B$

This answer is categorised as wrong, since it does not contain any accurate parts. The algorithm calls the strategy ***Socratic***, which produces the dialogue-move *signal-wrong* (as general pedagogical feedback) and calls the subtask ***conceptual***, as no pragmatically interesting elements are found in the answer.

Since the number of hints produced so far is $LH = 0$, the number of wrong answers $LW = 1$ and since there is no domain knowledge relevant to the proof step expected[6], subtask ***proof-step-meta-reasoning*** is called through first subtask ***per-step*** and then subtask ***domain-object***.

Note that the search for domain knowledge is done by help of the domain ontology, represented in the HSS. The first applicable hint is produced, which tries to get the student to reason about what to do first, and the indication that this is a subtask. The instructional point is Proof Step and more specifically Starting Point. The hint is active, because the GMCLA $\leq 0.75$, *give-away-relevant-concept* is produced. The Relevant Concept is the implication, which is referred to as "the if-then relation". The active function of this hint is to elicit the Rule of Inference, as will become evident later on in the example.

**T2:** (`signal-wrong`) This is not quite right. (`initiate-subtask-proof-step-meta-reas`) Let's see now. (`elicit-starting-point`) First think of how can you start handling the problem?
**S2:** (`irrelevant`) $A \subseteq K(K(A))$

[6] As an example of domain knowledge here, the Relevant Concept is the implication.

With this hint, the $LH$ is incremented by 1. However, the hint is not sufficient to help the student. She gives an answer that is correct in principle, but does not lead anywhere. We call such answers `irrelevant` and they increase the $LW$. The same functions are called, but this time another hint is produced, which tells the student where the reasoning should start. This is a passive hint, hence a prompt is also produced.

**T3:** (`signal-irrelevant`) That's correct, but not really relevant at the moment. (`give-away-starting-point`) Identify rather what's given and what you have to prove. (`prompt`) Go on!
**S3:** (`resign`) Nein *[No]*

The new student answer is equivalent to *"I don't know"*, which we categorise as a *resign*, and increments the $LW$. Therefore, the function SOCRATIC produces an explicit *encourage*, which prompts the student to try to answer the previous hint, and reproduces that hint. The counters are now $LH = 2$ and $LW = 3$.

**T4:** (`encourage`) It's a bit difficult, right? (`prompt-resign`) But give it a go and see what you can do! Try to identify what's given and what you have to prove.
**S4:** (`unknown`) $A \subseteq K(B) \equiv B \subseteq K(A)$

The student answer cannot be categorised at all. Therefore the tutor points this out, closes the previous subtask and calls the ***diagnostics*** subdialogue, which first checks what the problem is.

**T5:** (`close-subtask-proof-step-meta-reas`) OK. (`signal-unknown`) I'm not sure I understand what you are getting at. (`initiate-subdial-diagnostics`) So, (`check-origin-problem`) can you explain a bit more?
**S5:** (`step-size` ) $A \subseteq K(B)$ daraus folgt $B \subseteq K(A)$
*[$A \subseteq K(B)$ from that it follows $B \subseteq K(A)$]*

The student's response to the clarification subdialogue is categorised as *step-size*, which means that what the student inputs is correct, but it needs to be proved. This however counts as a wrong answer in our model, and we now have $LH = 3$ and $LW = 4$. The tutor moves on with the reasoning, and asks the student to think about the premise and the conclusion.

**T6:** (`signal-step-size`) Right, that's quite a big step though. Do just (`elicit-prem-conc`) concentrate on what is assumed and what you have to prove first.

**S6:** (`wrong`) $A \nsubseteq B$

With this wrong answer, the student model is too bad to carry on hinting for this step. The tutor gives away any pending answers, the step and its meta-reasoning.

**T7:** (`signal-wrong`) OK! Now, that's not quite correct. (`give-away-prem-conc`) What we have to assume is that A is a subset of K(B) and we have to prove that B is a subset of K(A) (`close-subdial-diagnostics`) So, (`give-away-proof-step`) the proof step is: Let $A \subseteq K(B)$, we will show that $B \subseteq K(A)$. (`initiate-subtask-step-meta-reas`) Let's just look at the whole step now. (`give-away-proof-step-meta-reas`) We start by identifying what we have to assume and prove, here we have to assume that A is a subset of K(B) and prove that B is a subset of K(A). Then we concentrate on the *if-then* relation that is central to our mathematical expression and therefore helps us prove what we want. So, we use the inference *if x then y, assume x and prove y*, to get rid of the *if-then* relation. Finally, we substitute the *if-then* with this rule, which gives us the step: Let $A \subseteq K(B)$, will show that $B \subseteq K(A)$. OK, now (*prompt-step*) try the next step!

Now, the *LH* is again incremented by 1 and $LH = 5$, so we switch to the more guiding subtask ***`spell-out-task`***, which we do not depict here.

At the end of the session, the last tutor turn recapitulates the whole proof. We provide here the recapitulations of the proof that the tutor in the WoZ experiments formulated for participant *socratic23*, based on the proof steps that were actually used in this session.

**T-l:** (`initiate-subtask-recap-(proof)`) Ich zeige Ihnen nun den gesamten Beweis: Zunächst setzen wir die Gültigkeit von $A \subseteq K(B)$ voraus, denn dies ist die Voraussetzung. Dann nehmen wir ein beliebiges Element $x \in B$ und zeigen, daß dieses auch in $K(A)$ sein muß. Sei also $x$ in $B$. Dann ist $x$ nicht in $K(B)$ und deshalb nach Voraussetzung auch nicht in $A$. Wenn $x$ aber nicht in $A$ ist, so ist es in $K(A)$, was den Beweis abschließt, da $x$ beliebig war.

*[Now I'm going to show you the whole proof: First we assume the validity of $A \subseteq K(B)$, as this is the hypothesis. Then we take an arbitrary element $x \in B$ and show, that it has to be in $K(A)$ as well. Let $x$ be in $B$. Then $x$ is not in $K(B)$ and therefore according to the assumption also not in $A$. But if $x$ is not in $A$, it is in $K(A)$, which concludes the proof as $x$ was arbitrary.]* (`close-dialogue`) This task is now complete.

**Uncommented Example**

**T0:** (`initiate-dialogue`) Hello!
**S0:** Hello!
**T1:** (`initiate-task, prompt`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset K(B), then B is a subset of K(A)!
**S1:** (`wrong`) $A \subseteq B$
**T2:** (`signal-wrong`) This is not quite right. (`initiate-subtask-proof-step-meta-reas`) Let's see now. (`elicit-starting-point`) First think of how can you start handling the problem?
**S2:** (`irrelevant`) $A \subseteq K(K(A))$
**T3:** (`signal-irrelevant`) That's correct, but not really relevant at the moment. (`give-away-starting-point`) Identify rather what's given and what you have to prove. (`prompt`) Go on!
**S3:** (`resign`) Nein *[No]*
**T4:** (`encourage`) It's a bit difficult, right? (`prompt-resign`) But give it a go and see what you can do! Try to identify what's given and what you have to prove.
**S4:** (`unknown`) $A \subseteq K(B) \equiv B \subseteq K(A)$
**T5:** (`close-subtask-proof-step-meta-reas`) OK. (`signal-unknown`) I'm not sure I understand what you are getting at. (`initiate-subdial-diagnostics`) So, (`check-origin-problem`) can you explain a bit more?
**S5:** (`step-size` ) $A \subseteq K(B)$ from that it follows $B \subseteq K(A)$
**T6:** (`signal-step-size`) Right, that's quite a big step though. Do just (`elicit-prem-conc`) concentrate on what is assumed and what you have to prove first.
**S6:** (`wrong`) $A \not\subseteq B$

**T7:** (`signal-wrong`) OK! Now, that's not quite correct. (`give-away-prem-conc`) What we have to assume is that A is a subset of K(B) and we have to prove that B is a subset of K(A) (`close-subdial-diagnostics`) So, (`give-away-proof-step`) the proof step is: Let $A \subseteq K(B)$, we will show that $B \subseteq K(A)$. (`initiate-subtask-step-meta-reas`) Let's just look at the whole step now. (`give-away-proof-step-meta-reas`) We start by identifying what we have to assume and prove, here we have to assume that A is a subset of K(B) and prove that B is a subset of K(A). Then we concentrate on the *if-then* relation that is central to our mathematical expression and therefore helps us prove what we want. So, we use the inference *if x then y, assume x and prove y*, to get rid of the *if-then* relation. Finally, we substitute the *if-then* with this rule, which gives us the step: Let $A \subseteq K(B)$, will show that $B \subseteq K(A)$. OK, now (*prompt-step*) try the next step!

**T-l:** Now I'm going to show you the whole proof: First we assume the validity of $A \subseteq K(B)$, as this is the hypothesis. Then we take an arbitrary element $x \in B$ and show, that it has to be in $K(A)$ as well. Let $x$ be in $B$. Then $x$ is not in $K(B)$ and therefore according to the assumption also not in $A$. But if $x$ is not in $A$, it is in $K(A)$, which concludes the proof as $x$ was arbitrary. (`close-dialogue`) This task is now complete.

### 6.6.2 First and second proof steps with partial and correct answers

**Commented Example**

**T0:** (`initiate-dialogue`) Hello!
**S0:** Hello!

The HSS is empty, the tutoring task is initiated, and the student is prompted for the next step.

**T1:** (`initiate-task, prompt`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset K(B), then B is a subset of K(A)!
**S1:** (`partial-answer(ia)`) I have to identify what's given and what I have to prove.

This answer is a partial one, since what the student states is a correct subpart, but is not the Proof Step, as expected at this point. The local counts are $LH = 0$, $LW = 0$, $LC = 0$ and the GMCLA $\leq 0.3$. Subtask *proof-step-meta-reasoning* is called for the step. The student understands the starting point, so the active hint to address the Premise and Conclusion instructional points is produced.

**T2:** (encourage) Great! (signal-pa) You're on the right track. (initiate-subtask-proof-step-meta-reas) We're taking it from the start. So, go ahead and (elicit-prem-conc) find what is assumed and what you have to prove.

**S2:** (correct) I have to prove that B is a subset of K(A), and $A \subseteq K(B)$ is assumed.

The student response answers the eliciting hint, so it is correct, but the step is not completed, so the next hint elicits the proof direction. Note that the proof is direct, therefore the abstract method (Directness) is not addressed.

**T3:** (signal-accept) Correct! (elicit-specific-method) Now, how can you manipulate the expression to prove what you want?

**S3:** (correct) I have to simplify what we are trying to prove.

With this answer, the student has covered all proof-step meta-reasoning instructional points, so the subtask *rel-con-meta-reasoning* is called and since the GMCLA is still $\leq 0.3$ (there have been not more non-correct answers), the meta-reasoning for the Relevant Concept is elicited.

**T4:** (signal-accept) Correct! (close-subtask-proof-step-meta-reas) OK. (initiate-subtask-rel-con-meta-reas) Let's see, then. (elic-meta-reas-rel-con) Try to find something in the expression that would help you simplify the problem.

**S4:** (correct) Do you mean the if-then?

The student names the Relevant Concept correctly, and with that answer we move out of the meta-reasoning and the proof-step subtask *domain-object* is initiated, which asks for the Subordinate Concept.

**T5:** (signal-accept) Exactly! (close-subtask-rel-con-meta-reas) All right then. (initiate-subtask-domain-object) Now, (elicit-sub-concept) what do you know about the if-then relation that can help you manipulate the expression?

**S5:** (`correct`) Something about what I need to prove?

The Subordinate Concept is indeed what has to be proven, so the subtask ***`inference-rule`*** is called next, and produces the hint that just asks for the inference hint directly.

**T6:** (`signal-accept`) Exactly! (`close-subtask-domain-object`, `initiate-subtask-inf-rule`) So, (`elicit-inf-rule`) can you tell me then which rule you can use here?
**S6:** (`correct`) Yes, if X $\Rightarrow$ Y, then let X and prove Y.

The student knows the Rule of Inference, so the subtask ***`substitution`*** now just asks her to apply it.

**T7:** (`encourage`) Good, (`signal-accept`) that is correct! (`close-subtask-inf-rule`) OK. (`initiate-subtask-subst`) Now, (`elicit-substitution`) try to apply this rule to the expression.
**S7:** (`correct`) Let $A \subseteq K(B)$ and prove that $B \subseteq K(A)$.

The student applies the rule, and with that performs the first proof step. The tutor accepts it and prompts for the next step. All HSS apart from the GMCL are reset.

**T8:** (`signal-accept`) Correct! (`prompt-step`) Move on to the next step!
**S8:** (`correct`) Let x $\in$ of B, we will show that x $\in$ K(A)

This is the next Proof Step, so the answer is correct and the tutor just accepts it and prompts the student for the following step.

**T9:** (`signal-accept`) Correct! (`prompt-step`) What's the next step?
**S9:** (`wrong`) Also let y $\in$ of A

This is a wrong answer, so hinting starts from the ***`proof-step-meta-reasoning`*** subtask again, but this time leaving out the hint `starting-point`, since the student has already performed a step, and should know where to start. Instead, the premise and conclusion are elicited.

**T10:** (`signal-wrong`) That's not right, actually. (`initiate-subtask-proof-step-meta-reas`) Let's take it from the start again. First, (`elicit-prem-conc`) find what is assumed and what you have to prove.
**S10:** (`partial-answer`) x $\in$ B is assumed and I have to prove that B subset K(A)

The student cannot answer the eliciting hint, so the tutor provides the passive hint that gives the answer to the previous eliciting hint away.

**T11:** (`encourage`) OK. (`signal-pa`) Almost right. (`give-away-prem-conc`) What is assumed is x $\in$ B, and what you have to prove is x $\in$ K(A)
...

**Uncommented Example**

**T0:** (`initiate-dialogue`) Hello!
**S0:** Hello!
**T1:** (`initiate-task, promp`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset K(B), then B is a subset of K(A)!
**S1:** (`partial-answer(ia)`) I have to identify what's given and what I have to prove.
**T2:** (`encourage`) Great! (`signal-pa`) You're on the right track. (`initiate-subtask-proof-step-meta-reas`) We're taking it from the start. So, go ahead and (`elicit-prem-conc`) find what is assumed and what you have to prove.
**S2:** (`correct`) I have to prove that B is a subset of K(A), and $A \subseteq K(B)$ is assumed.
**T3:** (`signal-accept`) Correct! (`elicit-specific-method`) Now, how can you manipulate the expression to prove what you want?
**S3:** (`correct`) I have to simplify what we are trying to prove.
**T4:** (`signal-accept`) Correct! (`close-subtask-proof-step-meta-reas`) OK. (`initiate-subtask-rel-con-meta-reas`) Let's see, then. (`elic-meta-reas-rel-con`) Try to find something in the expression that would help you simplify the problem.
**S4:** (`correct`) Do you mean the if-then?

**T5:** (signal-accept) Exactly! (close-subtask-rel-con-meta-reas) All right then. (initiate-subtask-domain-object) Now, (elicit-sub-concept) what do you know about the if-then relation that can help manipulate the expression?

**S5:** (correct) Something about what I need to prove?

**T6:** (signal-accept) Exactly! (close-subtask-domain-object, initiate-subtask-inf-rule) So, (elicit-inf-rule) can you tell me now which rule you can use here?

**S6:** (correct) Yes, if X $\Rightarrow$ Y, then let X and prove Y.

**T7:** (encourage) Good, (signal-accept) that's correct! (close-subtask-inf-rule) OK. (initiate-subtask-subst) Now, (elicit-substitution) try to apply this rule to the expression.

**S7:** (correct) Let $A \subseteq K(B)$ and prove that $B \subseteq K(A)$.

**T8:** (signal-accept) Correct! (prompt-step) Move on to the next step!

**S8:** (correct) Let x $\in$ of B, we will show that x $\in$ K(A)

**T9:** (signal-accept) Correct! (prompt-step) What's the next step?

**S9:** (wrong) Also let y $\in$ of A

**T10:** (signal-wrong) That's not right, actually. (initiate-subtask-proof-step-meta-reas) Let's take it from the start again. First, (elicit-prem-conc) find what is assumed and what you have to prove.

**S10:** (partial-answer) x $\in$ B is assumed and I have to prove that B subset K(A)

**T11:** (encourage) OK. (signal-pa) You're almost right. (give-away-prem-conc) So, what is assumed is x $\in$ B, but what you have to prove is x $\in$ K(A)

...

### 6.6.3 Multiple steps and multiple substrategies

**Commented Example**

**T0:** (initiate-dialogue) Hello!

**S0:** Hello!

**T1:** (initiate-task, prompt) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset of K(B), then B is a subset of K(A)!

**S1:** (partial-answer) OK, prove B $\subseteq$ K(A) and simplify the if-then.

The student knows the Relevant Concept, that she has to deal with a direct proof, and that she needs to do a backward step. Therefore, the tutor asks for the Subordinate Concept in relation to the Relevant Concept.

**T2:** (`encourage`) Great! (`signal-pa`) You are on the right track. (`initiate-subtask-rel-con-meta-reas`) Next (`elicit-sub-con-meta-reas`) consider what you know about the if-then relation that can help you find the right rule for the next step?
**S2:** (`near-miss`) Do you mean what I have to assume?

The student writes “assume” instead of “prove”, and the two are *converse* to each other (cf. Chapter A). Here the student’s answer is categorised first as a near-miss, as it’s possible that the student really meant to write ”prove” instead of ”assume”. The student is doing well, so there is hope that she can correct her own mistake. Therefore, the tutor just points out the discrepancy.

**T3:** (`encourage`) Right. (`near-miss`) There’s a minor problem with your answer. (`discrepancy`) Is that really what you wanted to say? (`prompt-action`)
**S3:** (`wrong`) Yes.

The confirmation by the student makes her answer wrong, and by that the subtask *`near-miss`* is initiated, to treat the problem.

**T4:** (`close-subtask-rel-con-meta-reas`) OK, then (`signal-wrong`) that’s not quite right. (`initiate-subtask-near-miss`) Let’s see. (`give-away-converse`) What you said is the reverse of what you need. (`prompt-action`) Try once more!
**S4:** (`step-size`) A subset K(B) from that follows B subset K(A).

The student restates the expression to be proven (which was an actual step-size in the corpus, cf. Example 6.6.1 **S5**). Since there has been no other step-size, we don’t assume there’s a relevant misconception. However, the student also does not correct her answer based on the `give-away-converse` hint, so we consider that as a misconception. She might not be clear about the difference between premise and conclusion. The subtask *`misconception`* is called, which starts by asking the student to explain the problematic input a bit more. The student input is a parameter to the dialogue move, which `Menon` provides.

**T5:** (`close-subtask-near-miss`) OK. (`signal-step-size`) That is big a step. (`initiate-subtask-miscon`) Let’s see. (`check-origin-problem(converse)`) Can you explain first what you meant by “what I have to assume” above?

**S5:** (`step-size`) I meant that A is a subset of K(B) and that it follows that B is a subset of K(A).

The student insists on the step-size, although she was pointed to it. We don't treat the previous misconception just yet, as we have encountered another one already. Therefore, we now think there is a misconception relevant to the step-size. The student might not understand when something needs to be proven, so we ask her to explain that. The subtask *`misconception`* repeats this process until there are no further misconceptions encountered and then deals with all encountered misconceptions.

**T6:** (`signal-misc`) There seems to be some misconception here. Again, that step is too big. (`check-origin-problem(step-size)`) Can you explain it?
**S6:** (`partial-answer`) I have to use if-then and say let x and prove y.

The student names the Rule of Inference and seems to be out of the misconceptions cycle. We explain to her the misconceptions encountered via misconception pragmatic hints, and now we are over five hints, so we switch to subtask *`spell-out-task`* and elicit the Substitution.

**T7:** (`encourage`) Great! (`signal-pa`) you're making progress! (`close-subtask-explain-miscon`) So, (`misconception-converse`) keep in mind that what you assume is your premises and what you have to prove is your conclusion. (`misconception-step-size`) Also remember that you have to prove or justify everything that is not a premise, or not shown before. (`initiate-subtask-spell-out-task`) Now, (`elicit-substitution`) try to apply the rule let x and prove y to the expression.
**S7:** (`request-assistance`) How do I apply it?

The student doesn't know how to apply the rule and asks for explicit help. Because the purpose of the *`spell-out-task`* is to take them step-by-step through the task and she is going to be led to the application of the rule anyhow, the *request assistance* remains unanswered. Rather, we inform the student that this is a more guiding strategy and we insist that she answers the active hint produced before, and reproduce it.

**T8:** (`encourage`) It's a bit difficult, (`initiate-subtask-spell-out-task`) so let me help you do this step-by-step. (`elicit-substitution`) First see if you can apply the rule to the expression.
**S8:** (`request-assistance`) Which rule?

The student insists on asking for help, but this time she asks for the Rule of Inference, which is already known for this step. So, the tutor just gives it away with the pragmatic hint that indicates that it should be taken for granted, and insists again on eliciting the Substitution.

**T9:** (`encourage`) (`correct-info-(give-away-inf-rule)`) The rule if X $\Rightarrow$ Y, then let X and prove Y. (`elicit-substitution`) Can you apply it? (`prompt-action`) Can you apply it now?
**S9:** (`correct`) Let A subset K(B), then prove that B subset K(B).

The student seems to know how to substitute but instead of writing "B subset K(A)" she has written "B subset K(B)". This is considered an ill-formed near-miss to begin with, so the student is informed about it and the answer to the previous hint is provided, as we are inside ***spell-out-task***, before the student is prompted to write the step.

**T10:** (`encourage`) Good! Now, (`signal-nearMiss`) your answer is not exactly accurate, so think carefully. (`give-away-substitution`) You have to substitute the appropriate parts of your expression, for the variables in the rule. (`elicit-proof-step`) Now write the whole step!
**S10:** (`correct`) Let A subset of K(B), and prove that B subset K(A).

The student applies the rule correctly this time, and with this completes the step. Because so many hints were needed for this step, a recapitulation is produced and the hinting for the next step starts.

**T11:** (`signal-accept`) Correct! (`encourage`) Very good! (`step-meta-reas`) So, once again, the reasoning for this whole step is as follows: (`give-away-prem-conc`) We assume that $A \subseteq K(B)$ and what we have to prove is that $B \subseteq K(A)$. (`give-away-spec-meth`) Next, we want to simplify the expression. (`give-away-rel-con-meta-reas`) So, we start with the if-then relation, because it's central in the expression and can help us simplify the problem. (`give-away-dom-tech`) We have to get rid of the if-then, and (`give-away-elab-dom-obj`) the rule that helps us here is "if x then y, assume x and prove y". (`give-away-inf-rule-appl`) Finally, we substitute the variables in the rule by the corresponding values in our expression. (`elicit-start-point`) Now, how can you start attacking (`prompt-step`) the next step?

**S11:** (`partial-answer`) I have to look for a rule to handle this, right?

The student knows the abstract method, but nothing else about the step at this point. The student performance requires the meta-reasoning subtask to be called for more support, but it is good enough to yield eliciting hints. So, the tutor elicits the meta-reasoning for the Relevant Concept.

**T12:** (`encourage`) Right! (`signal-pa`) That's a good start (`initiate-subtask-rel-con-meta-reas`) Now, (`elicit-rel-con-meta-reas`) what can you see in the expression that might help you simplify the problem?

**S12:** (`correct`) The subset.

The Relevant Concept is provided, which makes the response correct. The tutor accepts it and elicits the meta-reasoning for the Subordinate Concept.

**T13:** (`signal-accept`) Correct! (`close-subtask-rel-con-meta-reas`) OK then. (`initiate-subtask-sub-con-meta-reas`) So, (`elic-sub-con-meta-reas`) what is connected to the subset and can help you prove what you want.

**S13:** (`wrong`) The K.

The student has a problem with the Subordinate Concept, so this is given away and the student is prompted for the next action.

**T14:** (`signal-wrong`) That's not quite right. (`give-away-sub-con-meta-reas`) You can consider the concept *element* and how it connects to the subset to manipulate the expression. (`prompt-action`) Can you move on?

**S14:** (`correct`) I got it! All elements of B should also be elements of K(A)

This is enough for the student to complete the step, but the student performance is not good. To make sure that the reasoning is understood, the tutor starts the ***`aligning`*** subtask, and reproduces the active equivalent of the first hint for this step.

**T15:** (`signal-accept`) Correct! (`close-subtask-sub-con-meta-reas`) So, (`initiate-subtask-aligning`) let us look at our steps once again. (`align-elicit-rel-con-meta-reas`) Which is the concept from which we start simplifying the problem?

**S15:** (`correct`) With the subset.

The answer is correct, so the teacher moves on with the next hint already produced, as part of ***`aligning`***, with a different phrasing.

**T16:** (`signal-accept`) Correct! (`align-elicit-sub-con-meta-reas`) And what was connected to the subset that can help you prove what you want?

**S16:** (`resign`) Yeah, yeah...

The student, still does not know the answer to this hint. Since we are inside the subtask ***`aligning`***, the tutor just gives this information away. There have been no other hints for this step, so no elicit align hint is produced for this turn and aligning for the step is now completed. The tutor prompts for the next step.

**T17:** (`encourage`) OK, then (`align-give-away-sub-con-metareas`) As we said, you need to consider the element and how it connects to the subset to manipulate the expression. (`prompt-step`) Now, what's the next step?

...

### Uncommented Example

**T0:** (`initiate-dialogue`) Hello!
**S0:** Hello!
**T1:** (`initiate-task, prompt`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset of K(B), then B is a subset of K(A)!
**S1:** (`partial-answer`) OK, prove B $\subseteq$ K(A) and simplify the if-then.
**T2:** (`encourage`) Great! (`signal-pa`) You are on the right track. (`initiate-subtask-rel-con-meta-reas`) Next (`elicit-sub-con-meta-reas`) consider what you know about the if-then relation that can help you find the right rule for the next step?
**S2:** (`near-miss`) Do you mean what I have to assume?
**T3:** (`encourage`) Right. (`near-miss`) There's a minor problem with your answer. (`discrepancy`) Is that really what you wanted to say? (`prompt-action`)
**S3:** (`wrong`) Yes.
**T4:** (`close-subtask-rel-con-meta-reas`) OK, then (`signal-wrong`) that's not quite right. (`initiate-subtask-near-miss`) Let's see. (`give-away-converse`) What you said is the reverse of what you need. (`prompt-action`) Try once more!
**S4:** (`step-size`) A subset K(B) from that follows B subset K(A).
**T5:** (`close-subtask-near-miss`) OK. (`signal-step-size`) That is big a step. (`initiate-subtask-miscon`) Let's see. (`check-origin-problem(converse)`) Can you explain first what you meant by "what I have to assume" above?
**S5:** (`step-size`) I meant that A is a subset of K(B) and that it follows that B is a subset of K(A).
**T6:** (`signal-misc`) There seems to be some misconception here. Again, that step is too big. (`check-origin-problem(step-size)`) Can you explain it?
**S6:** (`partial-answer`) I have to use if-then and say let x and prove y.
**T7:** (`encourage`) Great! (`signal-pa`) you're making progress! (`close-subtask-explain-miscon`) So, (`misconception-converse`) keep in mind that what you assume is your premises and what you have to prove is your conclusion. (`misconception-step-size`) Also remember that you have to prove or justify everything that is not a premise, or not shown before. (`initiate-subtask-spell-out-task`) Now, (`elicit-substitution`) try to apply the rule let x and prove y to the expression.
**S7:** (`request-assistance`) How do I apply it?

**T8:** (`encourage`) It's a bit difficult, right? (`initiate-subtask-spellout-task`) so let me help you do this step-by-step. (`elicit-substitution`) First see if you can apply the rule to the expression.

**S8:** (`request-assistance`) Which rule?

**T9:** (`encourage`) (`correct-info-(give-away-inf-rule)`) The rule if X $\Rightarrow$ Y, then let X and prove Y. (`elicit-substitution`) Can you apppy it? (`prompt-action`) Can you apply it now?

**S9:** (`correct`) Let A subset K(B), then prove that B subset K(B).

**T10:** (`encourage`) Good! Now, (`signal-nearMiss`) your answer is not exactly accurate, so think carefully. (`give-away-substitution`) You have to substitute the appropriate parts of your expression, for the variables in the rule. (`elicit-proof-step`) Now write the whole step!

**S10:** (`correct`) Let A subset of K(B), and prove that B subset K(A)

**T11:** (`signal-accept`) Correct! (`encourage`) Very good! (`step-meta-reas`) So, once again, the reasoning for this step is as follows: (`give-away-prem-conc`) We assume that $A \subseteq K(B)$ and what we have to prove is that $B \subseteq K(A)$. (`give-away-spec-meth`) Next, we want to simplify the expression. (`give-away-rel-con-meta-reas`) So, we start with the if-then relation, because it's central in the expression and can help us simplify the problem. (`give-away-dom-tech`) We have to get rid of the if-then, and (`give-away-elab-dom-obj`) the rule that helps us here is "if x then y, assume x and prove y". (`give-away-inf-rule-appl`) Finally, we substitute the variables in the rule by the corresponding values in our expression. (`elicit-start-point`) Now, how can you start attacking (`prompt-step`) the next step?

**S11:** (`partial-answer`) I have to look for a rule to handle this, right?

**T12:** (`encourage`) Right! (`signal-pa`) That's a good start (`initiate-subtask-rel-con-meta-reas`) Now, (`elicit-rel-con-meta-reas`) what can you see in the expression that might help you simplify the problem?

**S12:** (`correct`) The subset.

**T13:** (`signal-accept`) Correct! (`close-subtask-rel-con-meta-reas`) OK then. (`initiate-subtask-sub-con-meta-reas`) So, (`elic-sub-con-meta-reas`) what is connected to the subset and can help you prove what you want.

**S13:** (`wrong`) The K.

**T14:** (`signal-wrong`) That's not quite right. (`give-away-sub-con-meta-reas`) You can consider the concept *element* and how it connects to the subset to manipulate your expression. (`prompt-action`) Can you move on?

**S14:** (`correct`) I got it! All elements of B should also be elements of K(A)

**T15:** (`signal-accept`) Correct! (`close-subtask-sub-con-meta-reas`) So, (`initiate-subtask-aligning`) let us look at our steps once again. (`align-elicit-rel-con-meta-reas`) Which is the concept from which we start simplifying the problem?
**S15:** (`correct`) With the subset.
**T16:** (`signal-accept`) Correct! (`align-elicit-sub-con-meta-reas`) And what was connected to the subset that can help you manipulate the expression?
**S16:** (`resign`) Yeah, yeah...
**T17:** (`encourage`) OK, then (`align-give-away-sub-con-meta-reas`) As we said, you need to consider the element and use it to manipulate the expression. (`prompt-step`) Now, what's the next step?
...

### 6.6.4 Backtracking to previous turns

This examble illustrates how `Menon` can deal with the student taking back what she did before. Although it is not realistic that students will change their mind so many times in a row, we consider it here as it is a good illustration of how such a complicated case can be accommodated.

#### Commented Example

**T0:** (`initiate-dialogue`) Hello!
**S0:** Hello!
**T1:** (`initiate-task, prompt`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset K(B), then B is a subset of K(A)!
...
**S4:** (`partial-answer`) I have to use if-then and say that if, let x and prove y.

We assume here the proceeding turns and we are at the point where the Proof Step has been given away by the tutor after the partial answer by the tutor. The subtask *`proof-step`* is called.

**T5:** (`encourage`) Great! (`signal-pa`) You're on the right track. (`initiate-subtask-proof-step-meta-reas`) Now, let's see. (`give-away-proof-step`) The whole step is : Let A $\subseteq$ K(B), we will show that B $\subseteq$ K(A) (`give-away-step-meta-reas-(prem-conc, spec-meth, rel-con-meta-reas, elab-dom-obj, dom-tec, inf-rule-applic)`). The reasoning for this step is as follows: First, we assume that $A \subseteq K(B)$ and we have to prove that B $\subseteq$ K(A). Then we have to simplify the expression. We concentrate on the if-then relation, which can help us simplify the expression and prove what we want. We try to think of a rule that will help us eliminate the if-then, so we apply the rule if X $\Rightarrow$ Y, then let X and prove Y to the current step. (`prompt-step`) What's the next step?
**S5:** (`undo-proof`) I want to start over.

The student wants to start a new proof, therefore the whole HSS is reset apart from the GMCL and the tutor just accepts the student's wish.

**T6:** (`prompt-action`) Go on, then.
**S6:** (`partial-answer`) As I said before, I have to use if-then and say let x and prove y.

The student repeats the last answer **S4**, and since the GMCL is the same, the tutor also repeats the feedback given in **T5**, formulated differently to point out the repetition.

**T7:** Again, (`signal-pa`) you're thinking is good, (`initiate-sub-task-proof-step-meta-reas`) but consider now the whole step...(as in **T5**)
**S7:** (`undo-step`) No, no, I'll start again with this step. (`partial-answer`) First I have to find out what's assumed and what I have to prove.

The student now wants to try another step and starts in the same turn by naming the Starting Point, which she realises from the step-meta-reasoning hint. The LMCL is reset, the student answer is partially accurate, the subtask *`proof-step`* is initiated, and the tutor asks for the Premise and the Conclusion.

**T8:** (`encourage`) Good, (`signal-pa`) but you're getting there. (`initiatesubtask-proof-step-meta-reas`) So, (`elicit-prem-conc`) what can you assume and what do you want to prove?

**S8:** (`partial-answer`) OK, prove $B \subseteq K(A)$ and break down the if-then... (`resign`) I don't know.

The proof is direct and the student knows both the Relevant Concept and the Domain Technique, but she actually resigns, at the end. This *resign* is ignored, as there is a *domain-contribution* in the same turn. The subtask changes now to ***domain-object*** and the tutor tries to elicit the Subordinate Concept to help in a structured way.

**T9:** (`encourage`) Good! (`signal-pa`) You're still on the right track. (`initiate-sub-task-sub-con`) Now, (`elicit-sub-concept`) what do you know about the if-then which can help you manipulate the expression?

**S9:** (`partial-answer`) The rule says to break down the if-then let x and prove y. (`request-assistance`) So, what do I have to prove here?

The student's statement does not answer the previous elicit hint, so it is not correct, but only a partial answer *domain-contribution* again, which names the Rule of Inference. That means that the subordinate concept is not needed any more to find the Rule of Inference, the subtask changes to ***substitution***, and the tutor just asks for the substitution of the rule.

**T10:** (`encourage`) Great! (`signal-pa`) You're close. (`close-subtask-sub-con`) Now, (`initiate-subtask-subst`) (`elicit-subst`) why don't you try to apply the rule to the expression.

**S10:** (`undo-turn`) No, I take that back. (`correct`) I need to think of what I have to prove to help me get the Rule of Inference.

The student takes back the last turn, and actually answers the hint that tried to elicit the Subordinate Concept. So, the subtask ***spell-out-task*** is loaded now, as the hint count has exceeded five, and the tutor asks for the Rule of Inference, which was taken back by the undo. From now on, the tutoring is more guiding.

**T11:** (`correct`) Correct! (`close-subtask-subst`) So, (`initiate-subtask-spell-out-task`) tell me (`elicit-inference-rule`) which rule you can use here.
...

**Uncommented Example**

**T0:** (`initiate-dialogue`) Hello!
**S0:** Hello!
**T1:** (`initiate-task, prompt`) OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset K(B), then B is a subset of K(A)!
...
**S4:** (`partial-answer`) I have to use if-then and say that if let x and prove y.
**T5:** (`encourage`) Great! (`signal-pa`) You're on the right track. (`initiate-subtask-proof-step-meta-reas`) Now, let's see. (`give-away-proof-step`) The whole step is: Let A $\subseteq$ K(B), we will show that B $\subseteq$ K(A) (`give-away-step-meta-reas-(prem-conc, spec-meth, rel- con-meta-reas, elab-dom-obj, dom-tec, inf-rule-applic)`). The reasoning for this step is as follows: First, we assume that $A \subseteq K(B)$ and we have to prove that B $\subseteq$ K(A). Then we have to simplify the expression. We try to think of a rule that will help us eliminate the if-then, so we apply the rule if X $\Rightarrow$ Y, then let X and prove Y to the current step. (`prompt-step`) What's the next step?
**S5:** (`undo-proof`) I want to start over.
**T6:** (`prompt-action`) Go on, then.
**S6:** (`partial-answer`) As I said before, I have to use if-then and say let x and prove y.
**T7:** Again, (`signal-pa`) your thinking is good, (`initiate-subtask-proof-step-meta-reas`) but consider now the whole step...(as in **T5**)
**S7:** (`undo-step`) No, no, I'll start again with this step. (`partial-answer`) First I have to find out what's assumed and what I have to prove.
**T8:** (`encourage`) Good, (`signal-pa`) you're getting there. (`initiatesubtask-proof-step-meta-reas`) So, (`elicit-prem-conc`) what can you assume and what do you want to prove?
**S8:** (`partial-answer`) OK, prove $B \subseteq K(A)$ and break down the if-then... (`resign`) I don't know.

**T9:** (`encourage`) Good! (`signal-pa`) You're still on the right track. (`initiate-sub-task-sub-con`) Now, (`elicit-sub-concept`) what do you know about the if-then which you can help you manipulate the expression.

**S9:** (`partial-answer`) The rule says to break down the if-then, let x and prove y. (`request-assistance`) So, what do I have to prove here?

**T10:** (`encourage`) Great! (`signal-pa`) You're close. (`close-subtask-sub-con`) Now, (`initiate-subtask-subst`) (`elicit-subst`) why don't you try to apply the rule to the expression.

**S10:** (`undo-turn`) No, I take that back. (`correct`) I need to think of what I have to prove to help me get the Rule of Inference.

**T11:** (`correct`) Correct! (`close-subtask-subst`) So, (`initiate-subtask-spell-out-task`) tell me (`elicit-inference-rule`) which rule you can use here.

...

## 6.7 Conclusions

`Menon` provides templates for NL formulations as most state-of-the-art systems [VanLehn, 2006]. Nonetheless, using such templates makes obvious how the possibility of the hint realisation is restricted and how the expressiveness of hints is affected.

As an alternative which supports flexible NL realisation, `Menon` implements tutorial feedback in the form of dialogue moves chosen through a complex teaching strategy. In the examples we saw in this chapter, there is a variety of possible NL realisations for the same dialogue move. For instance, *encourage*, *signal-(domain-contribution-category)* and *initiate-subtask* occur very often in the examples, but their NL formulation is adapted to the context and attention is paid to avoid constant repetition. The reader can thus get a feeling of the importance of this differentiation that a real NL generator can provide, given the flexibility that `Menon` allows. `Menon` always provides the subtask to be initiated as a parameter to *initiate-subtask*, and the category of the *domain-contribution* as parameter to *encourage* and *signal-(domain-contribution-category)*. A NL generator can thus adjust the phrasing for the task accordingly. This is already a means for taking care of issues of discourse coherence that assist the learning process [Moore, 1993]. Note, however, that our goal is to separate the tutoring from the dialogue and sentence-level specifications of tutorial feedback, but not to provide the complete specifications necessary for the NL generation of feedback. However, by defining this separation and automatically providing tutorial feedback and its specification at the tutoring level, we enable the flexible realisation through further dialogue-move definitions and sentence-level specifications. These specifications can make allowance for issues such as coherence, motivation and politeness, which we discussed in Chapters 1, 2, and 4. We claim that, since all of these elements play a role in learning, the complex tutorial feedback should be composed by integrating them. `Menon` deals with those elements that pertain to tutoring, but not with the natural language or dialogue aspects of it. However, it suggests dialogue moves as the unit where these elements can be integrated. Within this general approach, `Menon` implements the tutoring aspect and provides an infrastructure for the integration of the different elements of tutorial feedback.

# Chapter 7

# Evaluation

## 7.1 Introduction

As an evaluation of our Socratic strategy, we asked 5 evaluators to rate:

- `Menon`'s Socratic teaching strategy as a whole, and
- individual instances of the automatic feedback produced by `Menon` in particular tutoring situations.

We found that 4 out of 5 evaluators preferred `Menon`'s teaching strategy as a whole to the didactic strategy that was the "winner" of our Wizard-of-Oz experiment (cf. Chapter 1). An evaluation of how well `Menon`'s overall strategy serves our global tutorial goals (cf. Chapter 2) and Bloom's [Bloom, 1956] affective and cognitive levels gave average to very good results. Additionally, 55% of the evaluators' choices of individual instances of feedback favoured `Menon`'s feedback to equivalent feedback of the previous "winning" strategy. The evaluation of this individual feedback instances with regard to our global tutorial goals was in total very good.

## 7.2 Background

Our comparative Wizard-of-Oz experiment showed that the instantiation of the didactic teaching strategy was the best strategy and resulted in better learning. Moreover, it was far better than the preliminary Socratic strategy which at that point only aimed at collecting data for developing and implementing a full-scale Socratic strategy. Our aim in the evaluation presented here was to compare the "winning" strategy in the Wizard-of-Oz experiment, with the enhanced full-scale Socratic strategy, which `Menon` implements.

Our participants were mathematics teachers and mathematics students in the final stage of their studies. They were all educated and taught in Germany. In order to have a more reliable result, we tried for pluralism in the sample.

As such, the participants had different degrees of teaching experience. The major difference was that some participants had taught at schools for decades, and some had just finished their teaching training as part of their university studies and had experience from one-to-one tutoring. This difference in teaching experience reflected a variety of teaching styles as the younger generations are exposed to more modern theories of learning and teaching approaches during their university studies and training. Finally, participants differed in age and gender.

The materials in the study consisted of 7-point Likert-scale questionnaires. The scales included statements and the degree these statements were true of the participants, with 1 representing complete agreement and 7 complete disagreement. The evaluation was carried out electronically and communication between the experimenter and the participants took place through e-mail.

## 7.3 Description of the Evaluation

This evaluation had two facets: One was to get a first impression on how professionals assess `Menon`'s strategy as a whole. The second was to get feedback on possible improvements of the individual feedback produced by `Menon`.

We used questionnaires to collect demographic data on the participants, on their teaching style and pedagogical background, and on their perceived ability to carry the evaluation through. The evaluators saw two tutorial dialogues and were asked to rate some selected feedback. This feedback was only a sample of the possible feedback that `Menon` can provide, as `Menon`'s feedback is adaptive. However, what was important was to capture this adaptivity. We did this by presenting the feedback to be evaluated as part of the more general context of tutoring a proof. In this context, feedback has to adapt based on the student's performance and needs and based on the way the task is developing. In effect, participants were evaluating how well the feedback fits the context and, hence, the appropriateness of `Menon`'s feedback in the specific context.

A questionnaire was used to collect data on the evaluators' preferred individual instances of feedback and overall strategy. `Menon`'s feedback was presented in natural language. Therefore, when compiling the questionnaires, we were careful to distinguish between evaluating the content of the feedback, which `Menon` is responsible for and which we wanted to evaluate, from the natural language phrasing of the feedback. As we have discussed in previous chapters, we do not make any claims about the appropriateness of the phrasings, so it was important to make this distinction in order to be able to keep only the feedback which really evaluated the content. These were then evaluated based on six main aspects of `Menon`'s feedback. The overall strategy was additionally assessed for how well it serves Bloom's objectives on the affective and cognitive levels, which concern our domain. Since Bloom's objectives cover our tutorial goals to a large extent, we used them as a stricter test.

**Population:** We recruited two secondary-school mathematics teachers, two mathematics students just before they took their final exams, and one mathematician who had completed his final exams, but was not yet practising teaching in a school. Consistent with the difference in their teaching experience, their descriptions of their teaching styles also varied quite a lot, and included characterisations like “conservative”, “question-oriented”, and “defined by clear rules and expectations”. Only one of them stated an awareness of schema theory at all, and they were not much aware of cognitive load theory (median=2, var=2.7), but they were all more or less aware of motivation theory (median=5, var=2.5). They stated that they supported strongly the promotion of self-sufficiency (median=6, var=0.3), but they were only moderately familiar with learner-oriented teaching (median=5, var=0.7), and they also moderately stated that they base their teaching on the needs of the student (median=5, var=1.5). The perceived competence items gave a satisfactory result (median=5, var=1.39), meaning that the evaluators felt capable of ranking the feedback.

### 7.3.1 Experimental Design

**Hypothesis:** We hypothesised that the evaluators would prefer `Menon`'s feedback more often than that of the previous “winning” strategy. We also hypothesised that the evaluators would give `Menon`'s feedback better ratings for serving our tutorial goals and Bloom's objectives.

**Confounds elimination:** We foresaw a few confounds, which we tried to eliminate.

- The participants might evaluate the NL formulations of the feedback instead of its content, which is what `Menon` produces automatically. Therefore, we collected separate data on NL formulation and content and hoped to reveal such a bias if it occurred.
- The participants might evaluate the system interface, rather than the feedback itself. Therefore, we asked them to evaluate feedback generated by the system, but presented it as a Word document.
- The participants might be too biased by their own teaching style, therefore we collected data on their teaching styles to be able to factor out such biases in post-analysis.
- The participants might not be familiar at all with the pedagogical theories behind the tutorial goals for which the feedback is testing. Therefore we collected data on their familiarity with these pedagogical theories.

A restriction to the design of the questionnaire was that we could not randomise the order in which the feedback choices were presented. We called the didactic feedback *Alternative A* and `Menon`'s feedback *Alternative B.* Following these categories, we asked at the end for an overall comparison and evaluation of feedback Alternatives A and B. Therefore, we always presented the didactic feedback first, giving it a step ahead.

### 7.3.2 Experimental procedure:

The experiment consisted of 4 main phases, which we describe in the following.

**Pre-Evaluation** This phase had two subphases:

1. The participants read a short introduction on the goals and the structure of the study.

2. The participants filled in a questionnaire on demographic data, and on their pedagogical and mathematics background.

**Evaluation** This phase had various subphases, which we enumerate in the order they occurred.

1. The participants read instructions on filling in the evaluation questionnaires.

2. The participants were shown a worked-out example of the task that was tutored in the tutorial dialogues that the participants evaluated.

3. The participants were presented with two tutorial dialogues between a student and a tutor on a proof task. They were also shown various individual feedback alternatives for particular points in the dialogue and the proof. These alternatives represented the didactic feedback (*Alternative A*) and `Menon`'s feedback (*Alternative B*). The participants were asked first to state their preference between the two alternatives of feedback. They were then asked to evaluate it by responding to specific questions. These questions aimed at having the evaluators rank how well the feedback meets our tutorial goals.

4. When they were done with the evaluation of the individual feedback alternatives, they were asked to state their overall strategy preference, by choosing either *Alternatives A* or *Alternatives B* as a whole for the individual feedback fragments. The participants then evaluated their overall choice based on how well it fulfilled our global tutorial goals and Bloom's educational objectives.

**Post-Evaluation** In this phase, the participants were asked to fill in a post-questionnaire, which consisted of two parts:

1. The first part collected data on the previous experience of the participants with and their attitude towards our tutorial goals and pedagogical theories that underly them.

2. The second part asked the participants to report on their perceived competence to evaluate the feedback.

## 7.4 Materials

We now explain the motivation of the design of the experiment materials. The original materials in German and an English translation of the experimenters materials can be found in Appendix F.

**Participants' Instructions** The evaluators were told that they would evaluate tutorial feedback for set theory and that their evaluation would be used to find the best feedback for the tutoring situation each time. They were also given instructions on what they were going to evaluate and on how to evaluate it. These instructions included the worked-out example of the task that was tutored in the tutorial dialogues that were evaluated.

**Pre-Questionnaire:** We designed the pre-questionnaire so that the items in it would not reveal our hypothesis. Therefore, we refrained at this point from asking any questions that concerned our tutorial goals and our preferred pedagogical theories.

**Evaluation Proofs, Scales, and Items** We used two tutorial dialogues (cf. Appendix F) as the background of the feedback alternatives that were evaluated. Both dialogues dealt with the same proof, but with different tutoring situations to depict `Menon`'s behaviour for different student responses and different student levels and have a larger sample evaluated of the possible feedback that `Menon` can provide. In the first dialogue, the feedback was short and consisted of general pedagogical feedback, motivational feedback, where necessary, and hints. The feedback in the second dialogue also consisted of these basic elements, but developed over more turns and depicted the use of `Menon`'s substrategies to help the student solve the task. This was juxtaposed in both dialogues with standard didactic feedback that gives away the answer.

The proof task tutored in the dialogues was the same one we used in the Wizard-of-Oz experiment. This allowed us to use the original feedback that the human tutor gave in the Wizard-of-Oz experiment for the didactic condition, wherever the tutoring situation had also occurred in the dialogues that took place in the experiment. For all other tutoring situations, we created the didactic feedback ourselves, based on the definition of the strategy and including any extra features that our human tutor added to it in the Wizard-of-Oz experiment. The feedback consisted of two parts. The first part informed the student of the categorisation of her answer (wrong, correct, etc.). The second part either just prompted the student for the next step if the answer was correct, or gave away the answer and explained it, in any other case. We also tried to use NL formulations that were similar to the ones produced by the human wizard. To create the tutoring situations we used student answers from the Wizard-of-Oz experiment, where available, as well as virtual answers to simulate new tutoring situations. To produce `Menon`'s feedback that would be evaluated, we fed these student answers to `Menon` and recorded its output. This is similar to the evaluation design used by [DiPaolo *et al.*, 2004], with the exception that they used only virtual student answers, for lack of real ones. Finally, since `Menon`'s output consists of dialogue moves, we constructed NL formulations for this output based on the definitions of the dialogue moves.

The length of the questionnaire and the time it would require to complete it constrained us to evaluating certain features of the feedback only, due to the time-restrictions of the evaluators. We selected more instances of innovative feedback in order to get a first impression of how teachers like it. We also included a bit more conventional feedback for comparison. The feedback also differed in the number of dialogue turns it involved.

The evaluators were asked to rank the individual feedback instances of their preference based on five main aspects of `Menon`'s output:

1. promoting schema acquisition,
2. motivating the student,
3. domain content of hints,
4. feedback content in the dialogue context,
5. cognitive load alleviation.

They were additionally asked to complete open questions if they had personal suggestions on NL formulations or other aspects of the feedback. With these questions we aimed at collecting data for improving the feedback.

The evaluators also ranked their overall strategy choice in a 7-point Likert scale that included 6 constructs. The first four represented our general pedagogical goals, and were:

1. distant transfer,
2. near transfer,
3. implicit learning, and
4. self-sufficiency

*Distant transfer* was defined as helping the student to learn general problem solving techniques and be able to apply them to other domains. *Near transfer* was defined as the equivalent for similar problems in the same domain. *Implicit learning* was defined as acquiring schemata that are not explicitly taught. *Self-sufficiency* was defined as providing a better chance to find a solution alone and bringing the student to the point of solving problems alone later.

The other two constructs that were included in the questionnaire represented the affective and cognitive levels in Bloom's objectives. They consisted of multiple items each, which represented the sub-objectives in Bloom's taxonomy. The affective level included the sub-objectives receiving, responding, valuing, organising, and characterising. The cognitive level included the sub-objectives knowledge, comprehension, application, analysis, synthesis, and evaluation. Where necessary and to the extent that it was possible, we tried to adopt the items in our scale to our domain. At the same time we kept the items general enough to mirror Bloom's understanding of the objectives. For example, the sub-objective *application*, was expressed as follows: "The student is more likely to solve problems in situations by applying the domain knowledge, proving techniques and rules in different ways", where the specification of the knowledge ("domain knowledge, proving techniques and rules") referred to our domain. These educational objectives were the criteria that the evaluators used to rate their strategy of choice.

It should be noted that since the evaluators could only speculate on the learning effects of the feedback based on the defined tutoring goals, what this study evaluated was whether the feedback produced by `Menon` is closer to what the evaluators as educators would have used than that of the didactic strategy. This is no substitute to evaluating the learning effects directly, but can give indications on the appropriateness of the feedback and possible enhancement, before embarking on large-scale studies with students. Such studies are the only way to avoid the fallacies which teachers are often prone to called the "expert blind spot" [Nathan and Koedinger, 2000]. This phenomenon describes the common failure of teachers and researchers to assume the viewpoint of the student due to their expertise in the domain. As a consequence, they are often poor judges of what kinds of difficulties students face in learning a particular domain, as well as when and which kind of help students need.

**Post-Questionnaire:** In the post-questionnaire, we collected data on all aspects of the evaluators' background that we could not refer to before the study. We asked them questions on their pedagogical background and specific questions on their preferred teaching style that addressed our tutorial goals. We used the validated questionnaire by [Williams *et al.*, 1998; Williams and Deci, 1996] that we adopted to capture the participants' perceived competence to evaluate the feedback.

## 7.5 Results

### 7.5.1 Overall Choice of Strategy

Four out of five evaluators chose `Menon`'s feedback as their preferred overall strategy. The results of their evaluation of this strategy with regard to our tutorial goals can be seen in Table 7.1. Since only one evaluator chose the didactic strategy, it does not make sense to compare the medians of the two strategies. The evaluators chose as the best characteristic of `Menon`'s overall strategy that it promotes distant transfer, near transfer, and self-sufficiency, and that it would have a positive influence on the students' affective state. Implicit learning scored a bit more than average, and Bloom's cognitive level scored exactly average.

| **construct** | distant transfer | near transfer | implicit Learning | Self-Sufficiency | affective level | cognitive level |
|---|---|---|---|---|---|---|
| **median**$_{men}$ | 5 | 5.5 | 4.5 | 5 | 5.5 | 4 |
| **var**$_{men}$ | 0.67 | 0.92 | 1.67 | 1.87 | 1.79 | 1.77 |

Table 7.1: Evaluation of `Menon`'s overall strategy

### 7.5.2 Choice of Individual Feedback

On the whole, evaluators were asked to choose between 8 individual feedback instances (4 one-turn and 4 multiturn ones) and to evaluate them. Since we had 5 evaluators, an aggregate of 40 choices was made. Out of these 40 choices, 22 choices (55%) favoured `Menon`'s output and the minority 18 choices (45%) favoured the feedback that simulated the "winning" didactic strategy in the Wizard-of-Oz experiment. Running an inter-rater agreement test per feedback choice and evaluation item was not appropriate because there were 5 evaluators rating on a 7-point Likert scale. For the same reason, it was not surprising that we found no significant results when comparing the scores on individual evaluation items across all feedback choices and evaluators. We will now look into more specific results for each feedback choice. These should be read with caution; however, they provide valuable impressions on the appropriateness of `Menon`'s feedback and directions for possible enhancements.

Table 7.2 is a summary of the results. It shows every one of the individual feedback choices. We represent the individual feedback choices by the name of the main feedback that `Menon`'s alternative delivered. Namely, either the name of the hint, for one-turn feedback, or the name of the subtask, for multiturn feedback. One-turn feedback was provided in the first tutorial dialogue and multiturn feedback in the second tutorial dialogue. The exact Socratic feedback and the alternative didactic feedback for every feedback choice can be found in Appendix F. The variance is shown where more than one evaluators chose this feedback, or where the construct consisted of more than one item in the questionnaire. These medians and variances are calculated for the sum of scores that one feedback got when it was chosen. The table shows the median and variance for `Menon`'s feedback (**median**$_{men}$, **var**$_{men}$) and the median and variance for the didactic feedback (**median**$_{did}$, **var**$_{did}$) for every construct (e.g. schema promotion) in every feedback choice. The most important of the results depicted in Table 7.2 are highlighted. We discuss the results in the following section.

#### 7.5.2.1 Results for the First Tutorial Dialogue

The first output tested if the evaluators would prefer hinting at the beginning of the session, rather than giving away the answer, even though the hint given addresses the meta-reasoning and starts with the basics of proving, namely identifying the premise and the conclusion. All participants preferred `Menon`'s

| | | **one-turn hints** | | | | **multiturn subtasks** | | | |
|---|---|---|---|---|---|---|---|---|---|
| *construct* | **individual feedback** | **elic-prem-conc** | **elic-spec-meth** | **elic-inf-rule** | **elic-subst** | **misc-oncep** | **req-ass** | **rel-con-meta-reas** | **align-ing** |
| *schema* | median$_{men}$ | 5.5 | 4.5 | **5** | **4.5** | **5.5** | 5 | 4 | 5 |
| | var$_{men}$ | 1.56 | 0.5 | 2.25 | 2.21 | 0.92 | 1.5 | 0.97 | 2.25 |
| | median$_{did}$ | – | 5 | 3.5 | 3 | 4.5 | 4.5 | 5 | 5 |
| | var$_{did}$ | – | 1.93 | 3.07 | – | 2.17 | 4.25 | 1.58 | 1.87 |
| *formulation* | median$_{men}$ | **6** | 6 | 5.5 | **5** | 5.5 | **6** | 5 | 4.5 |
| | var$_{men}$ | 1.7 | – | 0.5 | 1 | 4.5 | 3 | 4 | 2.25 |
| | med$_{did}$ | – | 5 | 5 | 2 | 5 | 4.5 | 4.5 | 5 |
| | var$_{did}$ | – | 3.3 | 1 | – | 1.3 | 0.5 | 0.5 | 1 |
| *content* | median$_{men}$ | 5 | **6** | 5 | 5 | **6** | **6** | **6** | 4.5 |
| | var$_{men}$ | 1.3 | – | 0 | 1 | 0 | 1.33 | 4.33 | 2.25 |
| | med$_{did}$ | – | 3.5 | 6 | 6 | 5 | 4.5 | 4.5 | 5 |
| | var$_{did}$ | – | 10.25 | 0.33 | – | 0.33 | 4.5 | 0.5 | 0 |
| *cognit. load* | median$_{men}$ | **6** | 5 | **6.5** | **5.5** | 4 | 5 | 5 | 6.5 |
| | var$_{men}$ | 1.2 | – | 0.5 | 0.92 | 0.5 | 1.33 | 0.33 | 0.25 |
| | med$_{did}$ | – | 6 | 5 | 3 | 3 | **6.5** | **6.5** | 6 |
| | var$_{did}$ | – | 0.67 | 0.33 | – | 2 | 0.5 | 0.5 | 1 |
| *dial. context* | median$_{men}$ | **6** | 6 | 5 | 6 | **6.5** | 6 | 6 | 4 |
| | var$_{men}$ | 1.2 | – | 2 | 0.25 | 0.5 | 2.33 | 3 | 1 |
| | med$_{did}$ | – | 5.5 | 5 | 7 | 5 | 6 | 5 | **6** |
| | var$_{did}$ | – | 0.92 | 1 | – | 2.33 | 0 | 2 | 0.33 |
| *motivation* | median$_{men}$ | **6.5** | 5 | **6** | **6** | 3 | 4 | 6 | 6 |
| | var$_{men}$ | 0.40 | 2 | 1.33 | 1.41 | 0.67 | 1.2 | 0.8 | 2.25 |
| | med$_{did}$ | – | 6.5 | 4 | 2.5 | 4 | **6** | 6 | 5.5 |
| | var$_{did}$ | – | 2.5 | 4.17 | 0.5 | 1.5 | 3 | 0.67 | 0.67 |

Table 7.2: Evaluation of `Menon`'s individual feedback

output to the output of the human tutor in the previous "winning" strategy, despite the fact that `Menon`'s output addressed a very basic proof level. This was consistent with what we expected, as mathematicians are familiar with the notions of premise and conclusion. The highest scores for particular tutorial goals were for better formulation, limited cognitive load, better information in the dialogue context, and more motivation.

The second output included an instance of the meta-reasoning hint for the elicit-specific-method. We were trying to test if the evaluators would carry on hinting and if they would find the hint appropriate, especially since we were expecting that they would not be familiar with either schema theory, or the particular way of breaking down the deductive process that we employed. This was indeed verified by the analysis of the post-questionnaires. We chose the hint that we thought would be out of the ordinary and maybe too abstract. Not surprisingly, only one participant chose this hint. However, another evaluator commented that the reason her did not choose this hint was not due to its content, but due to the NL formulation, which is not what we were evaluating. The same evaluator criticised the didactic feedback for potentially demotivating the student. The greatest differences in score were for better content, where `Menon`'s feedback scored higher, and for motivation, where the didactic method scored higher. All other scores were comparable between the two strategies. The overall data showed that the evaluators would prefer to let students work alone at this point in tutoring and motivate them, since they seem to have a grasp of the proof. Therefore, in this case the didactic feedback was chosen, which just accepted the answer and prompted for the next step. A comment also pointed out that `Menon` should provide more articulate motivational feedback. Rather than just "Good!", it should also explain how the "good" answer contributes to the task.

The hint given in the third feedback dealt with the Rule of Inference. Two out of three evaluators chose it, although the student had already received more hints than standard teachers normally give. The hint got mostly higher scores, for example for schema promotion, limited cognitive load, and for motivation. The rest of the scores are comparable.

The fourth feedback asked the student to apply the Rule of Inference, which she knows already. Four out of five evaluators chose this feedback. The highest differences in scoring were in favour of `Menon`'s output, and concerned schema promotion, better formulation, limited cognitive load, and motivation.

#### 7.5.2.2 Results for the Second Tutorial Dialogue

The feedback in the second tutorial dialogue tested the choice between using a substrategy or giving away the proof step. We were hoping to uncover the evaluators' attitude towards substrategies that consist of many turns and require the student to think hard. The results in this example were mixed.

The first feedback in this dialogue is given because the student restates the expression to be proven and it is not clear if she knows that she has not proved it. The subtask *`misconception`* is called to deal with this situation. It extends over several turns until it transpires that the student is really not aware of the fact that she has to prove everything that is not part of the assumption. Two evaluators preferred this feedback. It received high scores for promoting schema acquisition, for its content, and for appropriateness of information in the dialogue context. The scores were lower for cognitive load and for motivation. A critique of the didactic feedback by an evaluator who actually chose it was that it gives too much information away and does not help the student move on with the task, whereas the evaluator praises `Menon`'s output for motivating the student. However, the evaluator criticised `Menon`'s feedback for giving too many directions.

The second feedback in this dialogue applies the substrategy *`spell-out-task`*, which is triggered because the number of hints produced already is more than five. This substrategy leads the student step-by-step to completing the proof step. Hints are still chosen based on the student's demonstrated skill, but hinting will not stop until the step is completed. Finally, because many hints were needed for this step, a recapitulation is produced before the hinting for the next step starts. Three evaluators chose this feedback as opposed to just giving the step away. Highlights in scoring for `Menon`'s feedback included better formulation and content. On the contrary, the didactic feedback got better scores for limited cognitive load and motivation. One evaluator expressed a worry that Menon's feedback was too motivating, presumably meaning that the student might feel patronised.

The third feedback in this proof calls two meta-reasoning substrategies for the Relevant Concept and for the Subordinate Concept. They elicit the meta-reasoning for the respective instructional points. The student finds the first, but the tutor decides to give the other one away and prompts the student to continue. Again, three evaluators preferred this feedback. Menon's feedback was rated higher for content, but the didactic feedback scored better on cognitive load. There were no other major differences.

The last feedback was an instance of the substrategy *aligning*, which makes sure that the student has understood the process and its substeps if there were many hints used for a step. Two evaluators chose this feedback. The scores on individual items were on the whole similar, with the exception of appropriateness in dialogue context, where the didactic feedback scored higher. It is also worth mentioning that one of the evaluators specifically praised Menon's feedback for promoting schema acquisition. This is actually the purpose of the *aligning* subtask.

## 7.6 Discussion

The most striking result of the evaluation was that 4 out of 5 evaluators preferred Menon's overall teaching strategy. Menon's goal is to promote schema acquisition as a means of of promoting learning. With this evaluation we were aiming at showing that teachers find our teaching strategy better for learning and that it can pass the test of more general and independently defined criteria than our tutorial goals alone. Therefore, we included Bloom's educational objectives as part of our evaluation scale. As it turned out, Menon's teaching strategy obtained high rankings for most of our tutorial goals, as well as as for the affective level in Bloom's taxonomy. These results taken together, reveal a positive assessment of our teaching strategy in general. The cases where Menon's individual feedback was not mostly preferred and the low rankings it got can be viewed as possible points of improvement of the current strategy.

As an overall observation for hinting, it seems like evaluators were skeptical about things that they were not already familiar with, so they chose and scored based on what seemed more conservative. This should have been expected as evaluators can only make predictions of learning effects on the kinds of feedback that they have used, or of kinds of feedback that is based on theories that the evaluators are at least are aware of.

There was a comment that Menon's feedback in general was too verbose, and another one that it is too motivating. This may explain the low scores that the feedback got for motivating the student to pay attention. It is also a valid point that students can be demotivated if asked to read long feedback messages [Anderson *et al.*, 1995]. However, it is not clear whether this comment was meant as a criticism for the multiturn feedback, or only for the shorter feedback. Moreover, surprisingly enough, it was the same evaluator who expressed the wish for more articulate motivational feedback. Still, the criticism about the length of the feedback should be taken seriously into consideration in the design of NL formulations, which the criticism is mostly relevant to.

We collected separate data on NL formulation and content and hoped to reveal a possible bias against Menon's feedback due to the NL formulations that we made up. What became obvious from the study was that the evaluators did not even realise that Menon's feedback formulations were made-up, as opposed to the didactic feedback that was produced by an experienced human tutor. On the contrary, in a few cases the made-up formulations got a better score than the ones produced naturally by the human tutor. There may be two explanations for this. The first is that the NL formulations were indeed better. We know, however, from a dry-run of the questionnaire that this was not the case, as the evaluator complained about the formulations not sounding natural. This leads us to a second hypothesis, that the participants under "formulation" understood the general way of expressing oneself and scored the formulation for whether it was appropriate to use with a student. Unfortunately, this does not allow accurate interpretations, but it may be regarded as a positive outcome for the way Menon uses dialogue moves to produce motivational feedback.

There was a comment that pointed out how the evaluator was negatively influenced against some feedback due to the previous response of the student, which made the student appear demotivated. The particular student input was constructed and it should be noted that when constructing it we were not aware of this subtlety. This observation emphasises the importance of natural language not only in the output, but also in the input, as it is the medium of such subtleties that are otherwise not registered.

One evaluator made it clear that tutoring the first step of the proof should be skipped, and that the deductive process should not be broken down in such small steps, but rather only concentrate on tutoring the Rule of Inference. We conjecture that this would be possible for high level students who already have a schema that allows them to consider the first step trivial and for finding the Rule of Inference. However, such an approach does not do much for schema acquisition as it gives too much information away at once and still does not explain how the tutor came about this information. This evaluator expressed a worry that the feedback expected the student to guess what was on the teacher's mind. There are two sides to this. One concerns implicit learning. If the student is advanced enough to learn with little feedback, then the positive outcome is that implicit learning is promoted. On the other hand, if students are not that advanced, asking them to find the Rule of Inference without guidelines on how to do that is probably not going to help them with finding the rule alone next time. In fact, in this case the feedback is prone to the exact same criticism, that the teacher expects students to find out what she has in her mind, however without support this time. In addition, the expert blind point phenomenon [Nathan and Koedinger, 2000] should warn developers of the fact that teachers prefer to let students work more independently in cases where actually students learn better when they receive help. Nonetheless, Menon's output could be improved to accommodate better the student's level and take into account both of the issues discussed. This could be done, for instance, if hinting did not start immediately at the beginning of a session when the student input is a partial answer, as it currently does not start when the answer is correct. What one could conclude from the data in general is that the more demanding hints are the ones that elicit concepts that the evaluators are not familiar with, like the Relevant Concept and Subordinate Concept. Therefore, another appropriate improvement could be that we need to increase the threshold of the student level that permits eliciting such concepts and rather give them away more often.

The suggestion to provide acknowledgements with more content as responses to the students' correct answers can be addressed by acknowledging the correct parts explicitly, as proposed by Dzikovska and colleagues [Dzikovska *et al.*, 2008]. In fact, since we already represent the correct parts in terms of used instructional points the extension needed would be to add the instructional points as a parameter to the dialogue move *acknowledge* in `Menon`'s output. The NL generator could then realise an such explicit acknowledgement.

In terms of the overall strategy, all except for two aspects of `Menon`'s feedback were ranked high. The two aspects that were ranked just above average and just average were implicit learning and Bloom's cognitive level, respectively. The fact that implicit learning scored only just above average is consistent with the comments on the individual feedback choices, which complained about leading the student too much. This consistency is a good indication that despite the small sample the improvement that we discussed above is called for. This would allow students to work longer without hinting at the beginning of a session, or when they give partial answers, but their overall performance in the session is good. Such an improvement could also prove a good solution for finding the golden rule between breaking the tutored deductive process into too small and too big steps.

As far as Bloom's cognitive level is concerned, the low score it got for the overall strategy is a bit contradictory to the high score distant and near transfer got. Some of the sub-objectives of the cognitive level are equivalent to distant and near transfer, but are phrased more verbosely and are maybe not so clear. For instance, the item for distant transfer was phrased as "The student is more likely to apply what she learned in other domains as a general problem solving technique". The equivalent item in the cognitive level construct was phrased as "The student is more likely to make inferences and find evidence to support generalisations, analyse domain knowledge, relations and organisational principles". Arguably, the verbose description of the second item is only another way to refer to "a general problem solving technique". We also suspect that the verbose phrasings made the same points sound more demanding, which might have added to the evaluators' reluctancy to rank them similarly.

Moreover, Bloom's cognitive level is very widely defined, and it also includes sub-objectives that either do not apply well in our domain, or that the evaluators cannot assess because of their general nature. For example, the objective of promoting Synthesis requires students to present and defend opinions by making judgements about information and validity of ideas (e.g., judgements in terms of internal evidence). We did not manage to formulate this and similar objectives in a way that they would apply to our domain and goals, but they would still preserve Bloom's definition. Therefore, it is only natural that evaluators were reluctant to give high scores for such objectives. It is actually quite an achievement that Menon's feedback was ranked average in total for the cognitive level.

## 7.7 Conclusions

The major finding of this evaluation was that Menon's feedback was considered better than the didactic feedback that was the "winning" strategy in the Wizard-of-Oz study and was produced by a human tutor. This can only be seen as a tendency, as the sample was small, but it shows that the direction we have taken in the development of Menon appears to be the right one.

An unexpected result was that the NL formulations that we constructed for Menon's output were quite often given higher ranks than those of the human tutor. We suggested an interpretation based on which the evaluators were rather indicating a general agreement with the way the tutor (Menon in this case) expressed herself in the framework of tutoring. This in turn reflects a positive disposition towards the dialogue moves produced by Menon that define such ways of expression. For instance, dialogue moves that represent motivational feedback are part of what determines the appropriateness of the way a tutor expresses herself.

The data we collected can be used as pointers for Menon's feedback that could be improved in situations where the didactic feedback was chosen more frequently and got better results. The results of the ranking were not significant and our evaluators were not sufficiently aware of or trained in most aspects of the teaching style that we abide by. This fact may add to the objectivity of the evaluators, but is at the same time a reason for reading the results with caution. A study that would compare two alternatives, the old one and the new with corrections as indicated in Section 7.6, could clarify the issues that rose from the evaluation and provide stronger claims for possible improvements.

The next step would be a study that would compare learning effects of students. This study could also evaluate particular aspects of the feedback against each other. For instance, it could evaluate when it is better to provide meta-reasoning hints as opposed to performable-step hints, and the same for the rest of the hint dimensions, or when it is better to produce generic pedagogical feedback or call a substrategy. Such detailed evaluation is only possible because of the clarity of definition of Menon's strategy, which allows manipulation its feedback at this fine-grained level.

# Chapter 8

# Conclusion

We presented an approach to automating tutorial feedback and, in particular, the most complicated aspect of it, hinting. Our approach is an attempt to produce automatic feedback that is adaptive on two equally important levels: the dialogue and the tutoring level. We subdivided our investigation into a number of smaller research issues, each of which is summarised in the following.

From state-of-the-art research in learning science, we derived guidelines for instruction in teaching proofs. These guidelines and our observations from experimental data served as the basis for a teaching model which was then translated into a concrete Socratic teaching strategy. This Socratic strategy represents the controlled application of various pedagogical theories, as opposed to the common practice of most tutors, who do not consistently use such theories.

Looking at hints and their use in tutorial dialogue, we proposed a method for their automation. In particular, we separated task and discourse by defining a multidimensional dialogue move taxonomy which separates cognitive and dialogue function of hints. We investigated the cognitive functions of hints in more depth.

Our model of a tutoring session, the HSS, is organised into classes and subclasses which represent knowledge to estimate the students' level of understanding, their motivation and the cognitive load imposed on them. Each class or subclass is subdivided into fields and the aggregate of the values of the fields at each point define a tutoring situation. The Socratic strategy uses these tutoring situations to reason about appropriate automatic feedback.

We captured domain-content specifications for the hint categories in a separate dimension of the hint taxonomy, the domain knowledge dimension, which captures the abstract variables and relations that constitute our instructional points. We provided formalisations of the instructional points in a domain ontology for set theory. The definition of such instructional points was inspired by schema theory.

Various aspects of our approach are specifically designed to facilitate a dynamic natural language realisation of hints:

1. the definition of hints as dialogue moves in which we separate dialogue from cognitive functions

2. the specification of the domain content of hints in terms of abstract instructional points, which allows on-the-fly instantiation of this content for specific tasks

The constituents of hints represent different aspects whose choice dependents on the dialogue and tutoring context, a prerequisite for producing feedback that is dialogue and tutoring context adaptive.

Our tutorial manager `Menon` is a proof-of-concept implementation which automatically chooses the tutorial feedback and provides specifications for natural language generation of this feedback. The domain-content specifications are instantiated by a domain reasoner and are used along with the dialogue move definitions to bring together the different aspects of the tutorial feedback and discourse structure into a meaningful natural language piece of feedback. The domain content instantiation and the natural language generation of the tutorial feedback are not currently implemented, however, a prototype of the overall approach to tutorial dialogue management that integrates all aspects discussed here was built and evaluated as part of the DIALOG project (cf. Chapter 1), and more research in that direction has taken place since then (see, for example, [Tsovaltzi and Karagjosova, 2004], [Autexier *et al.*, 2004; Autexier and Fiedler, 2006]).

### 8.0.1 Domain Independence

An important aspect of our approach is that it can be used in other domains as well. To begin with, we explored data in another domain, namely basic electricity and electronics from a BEE corpus [Rosé *et al.*, 2001b] and applied the hint categories defined for the domain of set theory [Tsovaltzi, 2001]. A second important source for the definition of our hints was the research done in the domain of physiology and blood circulation in the CIRCSIM project [Hume, 1995]. Many elements of the strategy we implemented were motivated from this research. For example, the subtask *`spell-out-task`* was inspired by a tactic defined in the CIRCSIM project and was a strategy in our first attempt to define domain independent hint categories from the BEE corpus. Based on this idea, we defined the current subtask *`spell-out-task`*. A similar process gave rise to the subtask *`diagnostics`*. Moreover, out of the ten dialogue moves defined in the task dimension of our dialogue move taxonomy seven were already defined for the BEE corpus. The basic principle in our current definitions of the Relevant and Subordinate Concepts, for example, were also inspired by the BEE data. Out of a total of thirteen hint categories that were originally described based on this data, ten were retained and further defined for our new taxonomy, one was converted to a subtask (*`spell-out-task`*), one was analysed into further categories (the pragmatic level) and one – *encourage* – was newly defined as a task dialogue moves rather than as a hint. As far as the HSS is concerned, the basics of the operationalisation of the global and local HSS were identified already in the process of modelling the BEE tutorial dialogues and seven categories of our current domain-contribution categories were also identified but were actually defined in this thesis. For example, the category partial answer was defined through the definition of our instructional points. Furthermore, domain-independent characteristics were derived through comparison with the hints and Socratic algorithm that were designed in our attempt to model the phenomena observed in the BEE corpus. The distinction between dialogue vs. cognitive, pragmatic vs. conceptual and active vs. passive functions of feedback were already conceptualised in this first attempt.

Moreover, the domain knowledge dimension is domain dependent, everything else is not. Consequently, the definition of hints based on dimension choices is domain independent up to the choice of the instructional point to be addressed. The domain content defined by the choice of instructional points in the domain knowledge dimension is domain dependent.

Our work has also given rise to guidelines for structuring a domain in order to automate the domain-dependent content of hints, for example:

1. A justification for every step must be captured in an instructional point like the Rule of Inference.

2. If possible, the different justifications in a domain should be classified into more general domain techniques, as in our Domain Technique.

3. Instructional points that help the student identify the right justification to apply, like the Relevant and the Subordinate Concept, must also be defined.

4. The meta-reasoning relevant to the defined instructional points must also be captured in corresponding instructional points.

We expect that the greatest overlap will be for other mathematical domains, or other well-defined domains that have a similar structure and which often use theorem provers, for example problem solving in physics [VanLehn *et al.*, 2005]. More specifically, for a different proving domain, both the class structure and the dialogue move and hint dimensions can be kept the same, but the formalisation of the hint categories will have to be adapted. However, for a shift to a non-proving but still a problem-solving domain, there might be a need to add more classes, or more hint categories within the existing classes. The hint and dialogue dimensions will still be the same.

The hinting session status HSS, in turn, is a representation of the way the tutoring session evolves. It consists of the local HSS (LMGL), which represents domain contributions and other student dialogue moves for local evaluation, and of the global HSS (GMCL), which looks into the whole proof step and tutoring session. The student dialogue moves and the *domain-contribution* categories in the LMGL are not domain specific, but the formalisation of the *domain-contribution* categories is of course domain specific. The GMCLA is an aggregate of the values of a number of other fields in the HSS that is motivated by general domain-independent pedagogical considerations. On the whole, the structure of the HSS with its fields and their subfields can be retained as is, whereas only the *domain-contribution* categories must be defined for the new domain each time.

Additionally, our Socratic strategy makes no use of any domain-specific characteristics of the categories and moves that it employs and can thus be applied to a different problem-solving domain, especially the ones that have a structure similar to proving.

In summary, the enhanced ontology shows a way to capture domain notions. The hint taxonomy shows where these notions fit in terms of tutoring. The HSS operationalises the different guidelines of the tutorial model and allows defining tutoring situations for hinting. The implemented Socratic strategy shows how they can all be used for implementing the tutorial model but without reference to a specific domain.

## 8.1 Future Work

Our work is still a first attempt and the results from our evaluation are only indicative. There is room for improvement to the feedback that Menon offers at the moment, and more work needs to be done in the direction of automating feedback in general.

### 8.1.1 Tutoring Model Extensions

To start from our global tutorial goals, affect is represented in our system by basic criteria like how many times a student has asked for help, or if she wants to quit. To represent affective aspects more consistently and to capture more motivational issues, one could, for example, represent explicitly more dialogue moves like *resign* related to affect. Such dialogue moves may already be defined in our dialogue move taxonomy, however the fact that they are informative as to the affectional state of a student is not recognised and they do not have a function in the task dimension. Therefore, Menon does not deal with them at the moment. To make such additions a more precise inquiry must be assumed into the way affect manifests itself in general, but also with specific reference to natural language.

There is also empirical evidence that metacognition may be important to acquiring problem-solving skills. [Delclos and Harrington, 1991], for instance, found that monitored problem solving helps students reflect on their problem solving. They compared two conditions, one where students monitored their actions and one where they just solved problems. In particular, students in the monitoring condition were encouraged to pay attention to the strategies they used during problem solving in the training phase of the experiment. This increased their ability to transfer in the problem solving phase that followed. Aleven and colleagues [Aleven *et al.*, 2006; Roll *et al.*, 2006] have implemented metacognitive hints that help the students seek help at appropriate places. They used the Cognitive Tutors technology and implemented additional production rules that capture good and buggy help-seeking behaviour. These rules are based on three main criteria: (i) the time needed for deliberate action, (ii) the skills involved in a step and the probability that the student masters the skills, (iii) what the student has already done in a step, e.g., the number of successful and unsuccessful attempts and the number of previous hints. The buggy behaviour is clustered in four super-categories: (a) Help Abuse, (b) Help Avoidance (c) Try-Step Abuse and (d) Miscellaneous Bugs. The student's actions are then traced and hints are provided that prompt the student to exhibit the most appropriate help-seeking behaviour at every point. This might involve encouraging the student to request a hint, advising the student to slow down etc.

Before metacognition can be included in `Menon` a thorough investigation is needed to establish the relation between cognitive, metacognitive and affective functions, and the interplay between the inherent metacognitive function of hints and hints that aim only at reinforcing metacognition must be elucidated. The architecture of `Menon` enables extensions. In order to incorporate metacognitive feedback in `Menon`, three additions are necessary. First, a submodule should be added to the HSS to represent modelling metacognitive aspects in the tutoring session. Second, a new substrategy should be implemented to manage the choice of appropriate metacognitive feedback, based on tutoring situations defined by the new metacognitive aspects in the HSS. Third, the control strategy, Generic Tutoring, should be extended to incorporate when the metacognitive substrategy should be called.

### 8.1.2 Teaching Strategy Enhancements

A result from our evaluation was that the evaluators would have preferred a less intrusive strategy at the beginning of a tutoring session, which would allow students to think more for themselves. Since this critique was expressed for the same feedback that was also given low scores for motivating the student, one can suspect that students might be more motivated if they were allowed more freedom at the beginning of a session, or that they might feel lost. One option for taking the evaluators preference into account would be to give student more freedom at the start, but only after they have become familiar with how hinting works, e.g. after the first step.

Another relevant recommendation for improving our feedback suggested more verbose motivational feedback that would include indicating what the student has done right. A more accurate place in our implementation for dealing with that comment would be to provide a more detailed *signal_evaluation*, which is already part of every feedback turn. At the moment, *signal_evaluation* is restricted to informing students of the evaluation of their *domain-contribution*. A more detailed version of it should include informing the student what is right in the *domain-contribution*, as recommended by the evaluator, before hinting at what is wrong. There were also a couple of instances of this behaviour in the Socratic condition of the Wizard-of-Oz experiment. For example, when the student knows the Starting Point, the premise and the conclusion, the tutor can inform the student that this was the right thing to do first and then hint for the next instructional point. The feedback may look like this:

**T:** (`signal_evaluation`) "You are right to begin with identifying what you can assume and what you need to prove." (`elicit-rel-con`) "Can you now find the thing in the expression with which you can start working on the step?"

This behaviour could easily be automated in `Menon` by extending the specification of the move *signal_evaluation* to include the instructional points that the student has covered.

A general advancement in our system would be to give feedback on student requests other than the content of the existing hints. For example, an ill-formed *domain-contribution* is one where everything else is correct, but a part of it is ill formed. After such a domain contribution, the student might ask which part exactly of the answer is ill formed. In such cases, `Menon` currently only points out that there is something ill formed and then produces the correct formulation. We could extend `Menon` to be able to answer the student's request for the position where the ill formed syntax occurred by naming the part that was wrongly substituted. The equivalent addition can also be made with regard to a the *domain-contribution* wrong-linguistic-term. This is a *domain-contribution* where everything is correct except for a part where the wrong term has been used. In that case, it would be possible to respond to a student's request for the exact term that is expressed using wrong terminology. As we said in Example 6.4, when the task is to prove $A \cap B \in P((A \cup B) \cap (B \cup C))$, an ill-formed *domain-contribution* would be:
$P((A \cup B) \cap (B \cup C)) = PC \cup (A \cap B)$

At the moment, `Menon` would tell an advanced student that there is a problem with the syntax by producing the hint:

**T:** (`Elicit-ill-formed`) "What you wrote there is not completely correct syntactically. Can you correct it?"

If the student cannot correct the mistake, `Menon` would respond with the hint:

**T:** (`Give-away-ill-formed`) "The right way to write this is $P((A \cup B) \cap (B \cup C)) = P(C \cup (A \cap B))$".

The extention we are suggesting is that when the student after the first hint asks where the syntactic mistake is, the system can point to the right hand-side of the equation and give the student another chance to correct it. This extension is dependent on the analysis module to be able to provide the exact spot of the syntactic error. It is important for Socratic tutoring that students are not denied an opportunity to correct their own errors when possible. It is also important to ITSs in general to be able to deal with students' requests for help like human tutors.

We would also like to provide a better way to connect the hints between steps, so that the sense of continuity is increased and the student develops a better feeling of the reasoning. The hint `point-backwards` already provides this functionality, but only for the current step. However, if we could point students back to previous steps for how they dealt with meta-reasoning hints, in particular, this would not only avoid just repeating the same hint with different domain content, but it could also increase the understanding of the abstract schema. For example, if we have produced the hint `elicit-relevant-concept-meta-reasoning-(subordinate-concept)` for a previous step, we can point the student to the previous mention of it even though the hint specifications for the new step (the Relevant Concept and the Subordinate Concept) would be different. When it is produced for the first time, the hint could be phrased as:

> **T:** "Now, find something in the expression, which you can connect to the subset and can help you find the right rule for the next step.", where the "subset" is the Relevant Concept.

A `point-backwards` that takes as hint specification the hint produced in the previous turn (here `elicit-relevant-concept-meta-reasoning`), could be phrases as:

> **T:** "Remember there was a step where you also knew from which concept to start. What did we do then to help us find the right rule for the next step? "

However, a closer investigation of the benefits of this addition would have to be made first, in order to really provide such hints when it could be beneficial. It has been, namely, observed that too much looking backwards and searching for similarities may result in a split attention effect [Sweller, 1988] and can be detrimental for learning [Berry and Broadbent, 1988].

A further improvement to the system and the potential to capture individual needs of students would be to extend the HSS to keep track of which kind of hints a student responds better to. If this is represented, we can reason about which kind of hints to avoid and which to prefer.

### 8.1.3 Empirical Studies

In general, large-scale evaluations with students to test specific features are necessary. `Menon` can be used as a platform for testing the different aspects of the tutoring model and hence contribute to the empirical investigations in this direction. Maybe the most important contribution in this respect is the proof–of-concept implementation of schema theory which allows to scientifically test this theory and its role in learning. For example, [Lim *et al.*, 1996] stated a difficulty in explaining why students who were inclined to use a forward technique did not learn better, as would have been predicted by studies like [Chi *et* *al.*, 1982]. Our hypothesis would be that the students did not have the means-ends technique available to them, which is the actual way novices set out to learn in general [Owen and Sweller, 1985]. On the other hand, there was no guidance to help them apply a forward-technique, which presupposes that they already possess a relevant schema. Consequently, the students were at a loss. Using `Menon`, one could test the hypothesis that such students who don't have a schema for problem solving would benefit most from our hinting towards a forward problem-solving technique.

# Appendices

# Appendix A

# The Mathematical Theory in OMEGA and the Definitions of Relations

In this appendix we provide examples of the mathematical knowledge on set theory that is represented in OMEGA. This knowledge is in the form of natural deduction rules both for presentation purposes and also because our domain is very close to natural deduction. However, OMEGA represents this knowledge as abstract inference rules. We also outline the additional relations that were first defined for tutoring purposes for the DIALOG project Wizard-of-Oz experiment [Tsovaltzi and Fiedler, 2003b] and we then adapted and augmented for our enhanced model of automatic hinting presented in this thesis.

## A.1 Concepts for Set Theory

The following draws on *sets* and on *inhabitants* of sets. This is the list of concepts necessary for tutoring set theory:

| | |
|---|---|
| $\in$ : element of | $\notin$ : not element of |
| $\cap$ : intersection | |
| $\cup$ : union | |
| $\overline{\cdot}$ : set complement | |
| $\setminus$ : set difference | |
| $\subseteq$ : subset | $\not\subseteq$ : not subset |
| $\subset$ : strict subset | $\not\subset$ : not strict subset |
| $\supseteq$ : superset | $\not\supseteq$ : not superset |
| $\supset$ : strict superset | $\not\supset$ : not strict superset |
| $\mathcal{P}$ : powerset | |
| $\emptyset$ : empty set | |
| $\top$ : truth | $\bot$ : falsity |

## A.2 Inference Rules and Notation Examples

Let $E_s$ be the expected inference step in a particular proof, consisting of the premises $\varphi_1, \ldots, \varphi_n$, the conclusion $\psi$ and the rule P. Inferences in our domain usually have the form:

$$\frac{\varphi_1, \ldots, \varphi_n}{\psi}\mathrm{P}$$

Let $A, B$ be sets and let $x$ be an inhabitant. Then an example is the Union in Powerset: If $A \in \mathcal{P}(C)$ and $B \in \mathcal{P}(C)$, then $A \cup B \in \mathcal{P}(C)$, can be written in the form of an inference rule as follows:

$$\frac{A \in \mathcal{P}(C) \quad B \in \mathcal{P}(C)}{A \cup B \in \mathcal{P}(C)} \cup \mathcal{P}$$

These are the inference rules as defined in the underlying logic of OMEGA[The OMEGA group, url].

### A.2.1 Basic Deduction Rules

These are the basic deduction rules needed and are represented in the OMEGA ontology in the highest of the nested theories.

1. $\vee$-$I_R$ Disjunction Introduction, the rule is applied to the right: if $A$, then $A \vee B$.
2. $\vee$-$I_L$ Disjunction Introduction, the rule is applied to the left: if $A$, then $B \vee A$.
3. $\vee$-$E$ Disjunction Elimination: if $A \vee B$, and both $C$ holds under assumption $A$ and $C$ holds under assumption $B$, then $C$.
4. $\wedge$-$I$ Conjunction Introduction: if $A$ and $B$, then $A \wedge B$.
5. $\wedge$-$E_L$ Conjunction Elimination, the rule is applied to the left: if $A \wedge B$, then $A$.
6. $\wedge$-$E_R$ Conjunction Elimination, the rule is applied to the right: if $A \wedge B$, then $B$.
7. $\Rightarrow$-$I$ Implication Introduction: if assume $A$ and prove $B$, then $A \Rightarrow B$.
8. $\Rightarrow$-$E$ Implication Elimination: if $A \Rightarrow B$ and $A$, then $B$.
9. $=$-$I$ Equivalence Introduction: if we can prove $B$ under the assumption $A$ and we can prove $A$ under the assumption $B$, then we have $A = B$.
10. $=$-$E_R$ Equivalence Elimination, the rule is applied to the right: if $A = B$ and $A$, then $B$.

11. $=$-$E_L$ Equivalence Elimination, the rule is applied to the right: if $A = B$ and $B$, then $A$.

12. $\neg$-$I$ Negation Introduction: if assume $A$ and prove a contradiction, then $\neg A$.

13. $\neg$-$E$ Negation Elimination: if $A$ and $\neg A$, then $C$.

14. $2\neg$-$I$ Double Negation Introduction: If $A$, then $\neg\neg A$.

15. $2\neg$-$E$ Double Negation Elimination: If $\neg\neg A$, then $A$.

16. $\exists$-$I$ Existential Introduction: if $A[t \leftarrow x]$, then $\exists x A$.

17. $\exists$-$E$ Existential Elimination: if $\exists x A$, and we can prove that $A[a \leftarrow x] \Rightarrow B$ (where $a$ is new), then $B$.

18. $\forall$-$E$ Universal Elimination: if $\forall x A$, then $A[t \leftarrow x]$.

19. $\forall$-$I$ Universal Introduction: if $A[a \leftarrow x]$ (where $a$ is new), then $\forall x A$

### A.2.2 Definitions

These are the basic definitions included in Omega

*Definition of Element:* The *elements* of a set are its inhabitants: $x \in A$ if and only if $x$ is an inhabitant of $A$.

*Definition of Intersection:* The *intersection* of two sets is the set of their common elements: $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$.

*Definition of Union:* The *union* of two sets is the set of the elements of both sets: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.

*Definition of Subset:* A set is a *subset* of another set if all elements of the former are also elements of the latter: $A \subseteq B$ if and only if for all $x \in A$ follows that $x \in B$.

*Definition of Strict Subset:* A set is a *strict* subset of another set if the latter has at least one element more: $A \subset B$ if and only if $A \subseteq B$ and there is an $x \in B$ such that $x \notin A$.

*Definition of Superset:* A set is a *superset* of another set if all elements of the latter are also elements of the former: $A \supseteq B$ if and only if for all $x \in B$ follows that $x \in A$.

*Definition of Strict Superset:* A set is a *strict* superset of another set if the former has at least one element more: $A \supset B$ if and only if $A \supseteq B$ and there is an $x \in A$ such that $x \notin B$.

*Definition of Powerset:* The *powerset* of a set is the set of all its subsets: $\mathcal{P}(B) = \{A \mid A \subseteq B\}$.

*Definition of Complement:* Given a universal set $U$ and a subset $A$ of $U$, the *complement* of $A$ is the set of all members of $U$ that are not members of $A$: $K(A) = \{x|x \in U \wedge x \notin A\}$.

*Set Equality* If $A \subseteq B$ and $B \subseteq A$ then $A = B$.

*Set Difference:* $A \backslash B = \{x : x \in A \wedge x \notin B\}$.

### A.2.3 Theorems

These are examples of the theorems of the set theory that are represented in OMEGA.

1. $\subseteq$-*Ref* Introduction of a formula that expresses the reflexivity property of subset ($\subseteq$): for A in sets , $A \subseteq A$
2. $\subseteq$-*Comp* Introduction of set complements relative to a subset relation: if $B \subseteq A$, then $\overline{A} \subseteq \overline{B}$
3. $\subseteq$-*Diff* Introduction of set difference relative to a subset relation: if $A \subseteq B$, then $A = B \backslash C$, where $C = A \backslash B$
4. $\subseteq$-$\cap$-$I_{branch}$ Introduction of set intersection relative to a subset relation (branching). Rule is applied to the superset: if $A \subseteq B \wedge A \subseteq C$, then $A \subseteq B \cap C$
5. $\subseteq$-$\cap$-$E$ Elimination of set intersection relative to a subset relation. The rule is applied to the subsets.: if $A \subseteq B \cap C$, then $A \subseteq B \wedge A \subseteq C$
6. $\subseteq$-$\cap$-$I_L$ Introduction of set intersection relative to a subset relation. The rule is applied to the (Left) subset: if $B \subseteq C$, then $A \cap B \subseteq C$
7. $\subseteq$-$\cap$-$I_R$ Introduction of set intersection relative to a subset relation. The rule is applied to the (Right) subset: if $A \subseteq C$, then $A \cap B \subseteq C$
8. $\subseteq$-$\cup$-$I_L$ Introduction of set union relative to a subset relation. The rule is applied to the (Left) superset: if $A \subseteq C \wedge B \subseteq C$, then $A \cup B \subseteq C$
9. $\subseteq$-$\cup$-$I_R$ Introduction of set union relative to a subset relation. The rule is applied to the (Right) superset: if $A \subseteq C$, then $A \subseteq B \cup C$
10. $\mathcal{P}$-$I$ Introduction of the powerset: if $A \subseteq B$, then $A \in \mathcal{P}(B)$
11. $\mathcal{P}$-$I$ Elimination of the powerset: if $A \in \mathcal{P}(B)$, then $A \subseteq B$
12. $\cup$-$\mathcal{P}$ Union of Powersets: $\mathcal{P}(A) \cup \mathcal{P}(B) \subseteq \mathcal{P}(A \cup B)$.section
13. $\cup$-*Com* Commutativity of Union: $A \cup B = B \cup A$.
14. $\cup$-*Assoc* Associativity of Union: $(A \cup B) \cup C = A \cup (B \cup C)$.
15. $\cup$-*Distr* Distributivity of Union: $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

16. $\cap$-*Distr* Distributivity of Intersection: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

17. $\left.\begin{array}{l} \subseteq\text{-}\cup\text{-}I_R \\ \subseteq\text{-}\cup\text{-}I_L \end{array}\right\} = \subseteq\text{-}\cup\text{-}I_{com}$: if $A \subseteq B$ or $A \subseteq C$, then $A \subseteq B \cup C$

18. $\left.\begin{array}{l} \subseteq\text{-}\cap\text{-}I_R \\ \subseteq\text{-}\cap\text{-}I_L \end{array}\right\} = \subseteq\text{-}\cap\text{-}I_{com}$: if $A \subseteq C$ or $B \subseteq C$, then $A \cap B \subseteq C$

The result of collapsing the two rules in each of the set of rules 17 and 18 with the application of commutativity to one of them (i.e., with respect to $\cup$ and $\cap$ respectively) may be considered as one rule. Representing this as an explicit rule makes the rule available for the hinting process.

$\subseteq$-$\cap$-$I_{com}$ that refers to commutativity and applies to subset, is not to be confused with $\subseteq$-$\cap$-$I_{branch}$ that refers to branching, and applies to superset, that is, the application of the operation on both sides. The rule $\subseteq$-$\cap$-$I_{com}$ says: if $A \subseteq C$, then $A \cap B \subseteq C$; whereas $\subseteq$-$\cap$-$I_{branch}$ says: if $A \subseteq B$ and $A \subseteq C$, then $A \subseteq B \cap C$.

Since definitions, theorems or lemmata can all be written as inference rules, they are all considered inference rules.

### A.2.4 Definitions of Relations for Tutoring

We now present the relations for tutoring that were defined in view of the Wizard-of-Oz experiment, conducted in the context of the Dialog project. These relations were defined by comparing the definitions of concepts and inference rules in Omega with respect to common patterns [Fiedler and Tsovaltzi, 2005]. In particular, relations between mathematical concepts, relations between mathematical concepts and inference rules, and relations between concepts, formulae and inference rules were defined. In Chapter 3, we used these relations to define our instructional points. Figure 3.2 is an overview of the enhanced ontology. It depicts the instructional points with their subclasses and instances.

#### A.2.4.1 Relations

We now present the relations that were identified by looking at the Omega ontology.

**Relations between Mathematical Concepts** Let $\sigma, \sigma'$ be mathematical concepts.

**Antithesis:** $\sigma$ is in *antithesis* to $\sigma'$ if they are opposite concepts (i.e., one is the logical negation of the other).
Notation: antithesis$(\sigma, \sigma')$.

Examples: antithesis$(\in, \notin)$
antithesis$(\subseteq, \nsubseteq)$
antithesis$(\subset, \not\subset)$

antithesis($\supseteq, \not\supseteq$)
antithesis($\supset, \not\supset$)

**Duality:** $\cap$ is *dual* to $\cup$.

This relation can probably be generalised, but this is not necessary for the purposes of tutoring set theory.

**Converse:** $\sigma$ and $\sigma'$ are *converses* if for all $a, b$ holds that $a\sigma b \Leftrightarrow b\sigma' a$.
Notation: converse($\sigma, \sigma'$).

Examples: converse($\subseteq, \supseteq$)
converse($\subset, \supset$)

**Hypotaxis:** $\sigma$ is in *hypotaxis* to $\sigma'$ if and only if $\sigma'$ is defined using $\sigma$. Then, $\sigma$ is a *hypotaxon* of $\sigma'$, and $\sigma'$ is a *hypertaxon* of $\sigma$.
Notation: hypotaxon($\sigma, \sigma'$) and hypertaxon($\sigma, \sigma'$).

Examples: hypotaxon($\in, \subseteq$)
hypotaxon($\in, \supseteq$)
hypotaxon($\in, \cup$)
hypotaxon($\subseteq, \mathcal{P}$)

**Primitive:** $\sigma$ is a *primitive* if and only if there is no hypotaxon of $\sigma$.
Notation: primitive($\sigma$).

Examples: primitive($\in$)
primitive($\top$)

**Specialisation:** $\sigma$ is a *specialisation* of $\sigma'$ if and only if for all $x_1, \ldots, x_n$ holds that $\sigma(x_1, \ldots, x_n)$ implies $\sigma'(x_1, \ldots, x_n)$.
Notation: specialisation($\sigma, \sigma'$).

Examples: specialisation($\subset, \subseteq$)
specialisation($\supset, \supseteq$)

**Generalisation:** $\sigma$ is a *generalisation* of $\sigma'$ if and only if $\sigma'$ is a specialisation of $\sigma$.
Notation: generalisation($\sigma, \sigma'$).

Examples: generalisation($\subseteq, \subset$)
generalisation($\supseteq, \supset$)

**Relations between Mathematical Concepts and Rule of Inferences**
Let $\sigma$, $\sigma'$ be mathematical concepts, P be an inference rule, $s$ the *source* expression, that is the expression which P is applied on, and $t$ the *target* expression, that is the expression which we arrive at after the application of P on $s$. Note that although $s$ might coincide in some cases with the premises and $t$ with the conclusion of a step, this is not always the case. Source and target are used instead, as tutoring should be based on leading the student from the source to the target, as the tangible units for deriving the next proof step. For forward steps, the source are the premises and the target is the conclusion. The reverse holds

for backward steps (cf. Chapter 2 and Section 3.4.3). The following relations are defined:

**Relevance:** $\sigma$ is *relevant* to P if and only if P can only be applied when $\sigma$ is part of $s$ or $t$.
Notation: relevant-to$(\sigma, \mathrm{P})$.

Examples: relevant-to$(\subseteq, \subseteq\text{-}\cap\text{-}I_{com})$
relevant-to$(\cap, \subseteq\text{-}\cap\text{-}I_{com})$

**Dominance:** $\sigma$ is *dominant* over $\sigma'$ for rule P if and only if $\sigma$ appears in both in $s$ and $t$ , but $\sigma'$ does not. $\sigma'$ has to appear in either $s$ or $t$.
Notation: dominant$(\sigma, \sigma', \mathrm{P})$.

Examples: dominant$(\subseteq, \cap, \subseteq\text{-}\cap\text{-}I_{com})$
dominant$(\subseteq, \cup, \subseteq\text{-}\cup\text{-}I_{com})$

**Subordination:** $\sigma$ is *subordinate* to $\sigma'$ for rule P if and only if $\sigma'$ is dominant over $\sigma$ for rule P.
Notation: subordinate$(\sigma, \sigma', \mathrm{P})$.

Examples: subordinate$(\cap, \subseteq, \subseteq\text{-}\cap\text{-}I_{com})$
subordinate$(\cup, \subseteq, \subseteq\text{-}\cup\text{-}I_{com})$

**Relations between Concepts, Formulae and Rule of Inferences** Let $\sigma$ be a mathematical concept, P be an inference rule, $s$ the source expression, and $t$ the target expression.

**Occurrence State:** A relation that captures how the occurrence of an instance of $\sigma$ in $t$ and $s$ is influenced. There are two occurrence states:

**Insert:** $\sigma$ is *inserted* by rule P if and only if $\sigma$ occurs in $t$, but not in $s$.
Notation: inserts$(\sigma, \mathrm{P})$.

**Extract:** $\sigma$ is *extracted* by rule P if and only if $\sigma$ occurs in $s$ but not in $t$.
Notation: extracts$(\sigma, \mathrm{P})$.

Examples: If the rule *Set Equality*

$$\frac{A \subseteq B \qquad B \subseteq A}{A = B}$$

is applied backwards it inserts the $\subseteq$ to both targets $A \subseteq B$ and $B \subseteq A$, and extracts $=$ from the source $A = B$. If applied forwards, the rule inserts $=$ from the target $A = B$, and extracts $\subseteq$ from both sources $A \subseteq B$ and $B \subseteq A$.

Let also $E_x$ be a mathematical expression.

**Inversion:** Let P and P′ be two rules, $\varphi$ and $\varphi'$ the major premises (the non-trivial ones)[1] of P and P′, respectively, and let $\psi$ and $\psi'$ be the conclusions of P and P′, respectively. Then P is an *inverse* of P′ and vice-versa if and only if $\varphi = \psi'$ and $\varphi' = \psi$.

Notation: $\mathrm{inverse}(\mathrm{P}, \mathrm{P}')$.

Examples: $\mathrm{inverse}(\mathcal{P}\text{-}E, \mathcal{P}\text{-}I)$.

**In:** $\sigma$ is *in* $E_x$ if and only if $\sigma$ occurs in $E_x$.
Notation: $\mathrm{in}(\sigma, E_x)$.

Examples: $\mathrm{in}(\subseteq, A \subseteq B)$
$\mathrm{in}(\Rightarrow, A \subseteq K(B) \Rightarrow B \subseteq K(A))$

[1]The non-trivial premises are the ones that contain the Relevant Concept, which we define in Section 3.4.1.1

# Appendix B

# Menon's Interface Language

In the following, we use square brackets to designate an optional specification. The output specifications in **bold** and their analysis are the ones that are not provided by `Menon` as they pertain to domain reasoning (cf. Chapter 5).

| | |
|---|---|
| **Input** | ::= task |
| | \| strategy |
| | \| undo |
| | \| tut-goal-status |
| task | ::= a string for the task |
| strategy | ::= socratic |
| | \| didactic |
| undo | ::= turn |
| | \| step |
| | \| proof |
| tut-goal-status | ::= studTaskDM |
| | \| domCon |
| | \| instructionalPoints |
| | \| pragmaticInfo |
| | \| proofStatus |
| instructionalPoints | ::= relConU |
| | \| subConU |
| | \| domRelU |
| | \| iRU |
| | \| inverseRuleU |
| | \| techniqueU |
| | \| startPointU |
| | \| premiseU (the source expression) |
| | \| conclusionU (the target expression) |
| | \| directnessU |
| | \| directionU |
| pragmaticInfo | ::= difTheory |

| | |
|---|---|
| | \| sameDomInfo |
| | \| orderedListA |
| | \| unorderedListA |
| orderedListA | ::= boolean |
| unorderedListA | ::= boolean |
| difTheory | ::= difNextStep |
| sameDomInfo | ::= domInfoCat |
| domInfoCat | ::= domInfoCat |
| | \| whichRelCon |
| | \| whyRelCon |
| | \| whichSubCon |
| | \| whySubCon |
| | \| whichIR |
| | \| whyIR |
| | \| domTech |
| | \| basicKnow |
| | \| basicKnowRef |
| | \| illFormed |
| | \| howSubst |
| | \| whySubst |
| | \| whichProofStep |
| | \| whyProofStep |
| | \| startPoint |
| | \| premConc |
| | \| absMethod |
| | \| specMethod |
| | \| irrAssReq |
| proofStatus | ::= proofStepCompleted |
| | \| proofCompleted |
| proofStepCompleted | ::= boolean |
| proofCompleted | ::= boolean |
| studTaskDM | ::= STDM $\wedge$ [STDM specifications] |
| STDM | ::= reqEval |
| | \| reqAss |
| | \| resign |
| | \| timeout |
| STDM specifications | ::= assReq |
| | \| domConCat |
| assReq | ::= domInfoCat except irrAssreq |
| domConCat | ::= correct |
| | \| pa |
| | \| ci |
| | \| wrong |
| | \| unknown |
| | \| misc |
| | \| missBasicknow |

| | |
|---|---|
| | \| stepsize |
| | \| irrel |
| | \| illFormed |
| | \| wrongLingTerm |
| | \| nearMiss |
| | \| other |
| nearMiss | ::= antithesis |
| | \| duality |
| | \| conversion |
| | \| specialisation |
| | \| generalisation |
| | \| nearMissWLT |
| | \| nearMissIllFormed |
| relConU | ::= boolean |
| subConU | ::= boolean |
| domRelU | ::= domain relations |
| domain relations | ::= antithesis (all as defined in ontology) |
| | \| duality |
| | \| conversion |
| | \| hypotaxis |
| | \| specialisation |
| | \| generalisation |
| | \| primitive |
| iRU | ::= boolean |
| inverseRuleU | ::= boolean |
| techniqueU | ::= boolean |
| premiseU | ::= boolean |
| conclusionU | ::= boolean |
| startPointU | ::= boolean |
| directionU | ::= boolean |
| directnessU | ::= boolean |

| | |
|---|---|
| **Output** | ::= substrategy |
| | \| tutorTaskDM |
| | \| conTaskManDM |
| | \| conComManDM |
| substrategy | ::= substrCat ∧ [subsrtPar] |
| substrCat | ::= subtask \| subdialogue |
| subdialogue | ::= diagnostics |
| subtask | ::= metareas |
| | \| per-step |
| | \| recapitulation |
| | \| requset-assistance |
| | \| explain-misc |
| | \| spell-out-task |

| | |
|---|---|
| | \| aligning |
| | \| nearmiss |
| subsrtPar | ::= completeProof |
| | \| completeStep |
| tutorTaskDM | ::= TTDM ∧ [TTDM specifications] |
| TTDM | ::= hint |
| | \| domConEval |
| | \| checkOrProb |
| | \| prompt |
| | \| encourage |
| | \| align |
| conComManDM | ::= CMDM ∧ [TTDM specifications] |
| CMDM | ::= initiateDial |
| | \| closeDial |
| | \|initiateSubdial |
| | \| closeSubdial |
| conTaskManDM | ::= TMDM ∧ [TTDM specifications] |
| TMDM | ::= initiateTask |
| | \| closeTask |
| | \| initiateSubtask |
| | \| closeSubtask |
| hint | ::= all possible values of hint in HSS ∧ [TTDM specifications] |
| domConEval | ::= all domConEval cats |
| TTDM specifications | ::= pragmatic info |
| | \| instructional points |
| | \| studTaskDM |
| | \| promptStep (just string for generator) |
| | \| promptAction (just string for generator) |
| | \| task (provided by student model module) |
| | \| substask |
| | \| subdialogue |
| | \| domConCat (all values of domConCat in HSS) |
| | \| hint (for align, step-meta-reas and point-backwards) |
| | \| assReq (all possible values of assReq in HSS) |
| pragmatic info | ::= speak to answer info |
| | \| point to info info |
| | \| take for granted info |
| **speak to answer info** | ::= [list position] (the position of the missing element in a list) |
| | ∧ [list elements] |
| | ∧ [rest list elements] |
| point to info info | ::= sameDomInfo (the info the student is pointed back to) |

| | |
|---|---|
| take for granted info | \| misconception |
| | ::= **correctTerm** |
| | \| **correctInfo** |
| **correctTerm** | ::= correctLingTerm |
| | \| wrongLingTerm |
| misconception | ::= miscType |
| miscType | ::= step-size |
| | \| hypotaxis |
| | \| inversion |
| | \| primitive |
| correctInfo | ::= [hint] |
| instructional points | ::= domain objects |
| | \| domain relation |
| | \| rule of inference |
| | \| substitution |
| | \| proof step |
| **domain objects** | ::= relevant concept |
| | \| subordinate concept |
| domain relations | ::= domain-relation name \| domain-relation concept |
| domain-relation name | ::= antithesis (all as defined in ontology) |
| | \| duality |
| | \| conversion |
| | \| hypotaxis |
| | \| specialisation |
| | \| generalisation |
| | \| primitive |
| **domain-relation concept** | ::= antithetical concept |
| | \| dual concept |
| | \| junctive concept |
| | \| hypotactical concept |
| | \| specialisation concept |
| | \| generalisation concept |
| | \| primitive concept |
| **rule of inference** | ::= basic knowledge (as in HSS) |
| | \| rule of inference |
| | \| inversion (as in ontology) |
| | \| domain technique |
| **substitution** | ::= substitution (the proof step) |
| | \| correct formulation (for the ill-formed defined in HSS) |
| **proof step** | ::= starting point |
| | \| premise$^+$-conclusion$^+$ (the source expression/-s and the target expression/-s) |
| | \| directness |

| | |
|---|---|
| | \| direction |
| | \| proof step |
| **basic knowledge** | ::= basic knowledge reference ∧ the basic knowledge |
| **rule of inference** | ::= [rule of inference name] ∧ |
| | \| [formula] |
| **rule of Inference name** | ::= names of definitions |
| | \| theorems |
| | \| lemmata |
| | \| tautologies |
| **domain technique** | ::= occurrence state of defined constant |
| | \| occurrence state of quantifier |
| | \| occurrence state of connective |
| | \| case distinction |
| | \| induction |
| **directness** | ::= direct proof |
| | \| indirect proof |
| **direction** | ::= forward step |
| | \| backward step |

# Appendix C

# Dialogue Move Taxonomy

## C.1 Philosophy of Taxonomy

In this taxonomy, we focus on the separation of generic dialogue management phenomena, on the one hand, and the manipulation of genre- and domain-specific phenomena involved in modelling different teaching models and domains, on the other hand. The latter we view as only a subpart of the dialogue manager. In order to achieve that, we propose a dialogue-move taxonomy in the tradition of the multidimensional structure of DAMSL [Allen and Core, 1997] to capture the multifunctionality of utterances. This possibility is an advantage of DAMSL compared to other major proposals, such as the Verbmobil dialogue-act annotation scheme [Alexanderson et al, 1997]. The Verbmobil scheme represents a flat hierarchical structure with growing specificity towards the leaves. The core of the dialogue moves defined for Verbmobil is moreover specific to the task, namely appointment scheduling, which renders it hard to reuse the annotation scheme.

We mainly divert from other approaches in that we suggest an expanded TASK dimension for clearly separating out and defining the genre and domain specific characteristics of dialogue [Tsovaltzi and Karagjosova, 2004]. This further facilitates the original idea of the DAMSL attempt for reusability and reconfigurability, as it provides a better framework for capturing what is generic in dialogue management and reusable between genres, and, on the contrary, what is specific to the genre or even the different domains. The latter also makes modelling the genre, and the different domains in it more straightforward. We define dialogue moves using empirical data (cf. [Wolska *et al.*, 2004]), as well as based on previous taxonomies [Allen and Core, 1997; Core *et al.*, 2002]. We seek interrelations among the moves in the different dimensions.

We now briefly describe the DAMSL scheme and then analyse our particular adaptations to it.

### C.1.1 DAMSL dimensions

DAMSL allows multilevel annotation. It distinguishes four dimensions according to the unit's purpose and role in dialogue:

- Communicative status: It captures whether the utterance is intelligible and whether it was successfully completed, or uninterpretable, abandoned, self-talk, etc.
- Information level: It provides an abstract characterisation of the semantic content of the utterance:
  - Task: "Doing the task", i.e., utterances that advance the task.
  - Task-management: "Talking about the task", i.e., utterances that discuss the problem solving process or experimental scenario.
  - Communication management: "Managing the communication", i.e., conventional phrases that maintain contact, perception, and understanding during the communication process: greetings, closings, acknowledgements ("Okay", "uh-huh"), stalling for time ("Okay", "Let me see"), signals of speech repairs ("oops") or misunderstandings ("sorry?", "huh?").
  - Other-level.
- Forward-looking function: It characterises what effect an utterance has on subsequent dialogue and interaction.
- Backward-looking function: It captures the way the current utterance is related to the previous dialogue.

### C.1.2 Adaptations to DAMSL

We assume six dimensions instead of only four in DAMSL. More specifically, we make the DAMSL sublevels *task, task-management* and *communication management* separate dimensions of their own for more clarity. These sublevels belong to the *information level* in DAMSL. Turning them into separate dimensions is necessary in tutorial dialogues to capture the complex phenomena observed (cf. Section C.2.1).

We modified and extended the dialogue moves in DAMSL with moves from the BE&E annotation scheme [Core *et al.*, 2002], which was developed for the tutorial dialogue genre. The BE&E annotation scheme is based on DAMSL, however it does not distinguish all levels that DAMSL provides. The BE&E annotation scheme additionally provides additional dialogue moves derived from a tutorial dialogue corpus. We adopt some of these moves, but in order to cater for more tutorial dialogue genre- and domain-specific phenomena, we define new dialogue moves and group them in a separate task level, following DAMSL (cf. Chapter 4). The need for such a separation in dialogue systems became obvious by our tutorial dialogue corpus and has also been advocated by [Allen *et al.*, 2001b] who use a modular architecture for dialogue and task planning, for the domain of route planning.

## C.2 Overview of The Taxonomy

In this section, we describe the dimensions in the taxonomy and state their practical use. The dimensions task, conventional-task management, and conventional-communication management are scrutinised in Chapter 4. Here we examine the dimensions forward-looking, backward-looking, and communicative status.

### C.2.1 The Dimensions in the Taxonomy

The taxonomy features many different dimensions to capture the fact that an utterance produced by a speaker may have many functions (serve many purposes) at the same time, much in the DAMSL fashion [Allen and Core, 1997]. Every one of the dimensions represents one of these potential different functions. The actual detailed description of what the utterance's function is in the particular context (the dialogue at hand) corresponds to one of the categories of dialogue moves defined in the dimensions themselves. In every dimension, there are different levels of description of the dialogue moves. These are structured in classes and subclasses up to three different levels deep. The current function of an utterance is represented on the deepest level of subclasses, which inherit the properties of the respective superclasses. There are in total six dimensions in the taxonomy.

#### C.2.1.1 Forward-Looking Dimension

The forward-looking dimension characterises the effect an utterance has on the subsequent dialogue [Allen and Core, 1997]. In this dimension there are 11 classes of dialogue moves (DMs), some of them with subclasses. Most of the DMs are adopted from DAMSL with slightly modified definitions, two are taken from BE&E, and four are newly defined. An instance of a dialogue move in this dimension is *info-request*. We define it as an utterance that requests information that the speaker does not possess. This definition differs from the one provided in DAMSL, according to which any utterance that creates an obligation for the hearer to provide information is an *info-request*. However, in tutorial dialogues it is necessary to distinguish between questions the answer to which the speaker knows but does not want to give away, and questions to which the speaker really does not know the answer. This is especially important in the case where the speaker is the tutor. We consider only the latter to be *info-request*. For the former case, we adopt the dialogue move *diagnostic-query* defined in [Core *et al.*, 2002] as an utterance by which "the speaker is testing whether a listener knows a piece of information by asking him to supply the information" (p.4). An example of an *info-request* from our corpus is C.1 and of a *diagnostic-query* C.2.

(C.1) **T:** "Glauben Sie, dass Sie das nun beweisen müssen oder ist das eine Folgerung aus dem Obigen?" (soc23p)
*[Do you think that you have to prove that now, or does it follow from the above?]*

(C.2) **T:** "Wissen Sie, wie Sie diese Beziehung benutzen können?" (soc12k)
*[Do you know how to use this relation?]*

#### C.2.1.2 Backward-looking Dimension

The BACKWARD-LOOKING dimension captures how the current utterance relates to the previous discourse [Allen and Core, 1997]. Every move in this dimension has an antecedent, i.e., a dialogue move in the preceding discourse, which is affected by the backward move currently performed. Note that the antecedent need not be necessarily in the immediately preceding discourse, but can be more distant. A move can also have multiple antecedents.

Most of the dialogue moves in this dimension are also adopted from DAMSL. An exception is the category information-relation that DAMSL suggests as a possible aspect of the BACKWARD-LOOKING function but does not elaborate on. We define five *address* moves that apply when neither of the more specific dialogue moves in this dimension *accept*, *reject* and *answer* apply. The main motivation for having *address* moves is to distinguish between the aspects resolvedness vs. aboutness of answers to questions, following [Ginzburg, 1996]. Resolvedness and aboutness are relations between questions and answers. An answer resolves a question when the answer provides information that positively or negatively resolves the question [Larsson, 2002]. "About" is a relation that accounts for the range of information associated with a question [Ginzburg, 1994]. An *answer* move always resolves the question. If the question is followed by an utterance that does not resolve it, but is about it, we consider it as an *address-info-request*. For instance, the tutor's utterance in Example C.3 is not an *answer* according to the distinction resolve vs. about: it does not answer the student's question but rather refuses to answer it. However, it does address the question and realises, thus, an *address-info-request* move:

(C.3) **S:** "was ist K ( a )?" (min11d)
*[ what is K ( a )?]*
**T:** "Das kann ich nicht beantworten"
*[I cannot answer this]*

We modify the definition of *answer* as an utterance that complies with an *info-request* action in the antecedent and resolves it. An example of an *answer* move from our corpus is C.4 below, where the student gives an answer to the tutor's *info-request* and resolves it:

(C.4) **T:** "Ist das noch derselbe Lösungsweg wie in der vorigen Antwort?" (did10p)
*[Is this still the same solution as in the previous answer?]*
**S:** "Nein"
*[No]*

# C.3 The Dialogue Move Taxonomy

## C.3.1 Segmentation

The basic units that realise dialogue moves are sentences and coordinate clauses. In some cases, two or more clauses of one sentence that are coordinate may perform separate moves in one and the same dimension. For instance, the sentence in Example C.5 is segmented into two units, where the first clause is an *assert* and the second an *info-request*, both in the FORWARD-LOOKING dimension.

(C.5) **T:** "Das ist richtig," (*assert*) "aber wie bringt das den Beweis weiter?" (*info-request*) (soc1p)
*[This is correct, but how does it help with the proof?]*

However, a complex sentence can also realise one single move. Therefore, basic units can also be grouped together and perform a single dialogue move. Each unit realises at most one dialogue move per dimension.

## C.3.2 Forward-Looking Dimension

In this dimension there are 11 classes of dialogue moves, some of them with subclasses. Only direct (explicit) speech acts are assumed[1].

1. Statement
   (a) Assert
   (b) Reassert
   (c) Other-statement
2. Influencing-addressee-future-action
   (a) Open-option
   (b) Action-directive
3. Info-request
4. Diagnostic-query

[1] Decision trees for the dimensions FORWARD and BACKWARD-LOOKING can be found in Chapter 4 to help the reader disambiguate among the moves in these dimension.

5. Understanding-query
6. Committing-speaker-future-action
   (a) Commit
   (b) Offer
7. Conventional
   (a) Conventional-opening
   (b) Conventional-closing
8. Apologise
9. Gratitude
10. Signalling-emotion
    (a) Frustration
    (b) Satisfaction
11. Other-forward-function

#### C.3.2.1 Statement

Its primary purpose is to make explicit claims about the world. Intuitively, an utterance that can be followed by "That's not true" is a statement. Weak forms of statement such as hypothesising or suggesting that something might be true also belong to this class. Statements can be *asserts*, *reasserts*, or *other-statements* [Allen and Core, 1997].

**C.3.2.1.1 Assert** The speaker is trying to make the addressee adopt a belief by communicating a claim about the world [Allen and Core, 1997].

**Obligations** The hearer has to address it, that is, she cannot ignore it. Any utterance addressing an *assert* will discard this obligation [Kreutel and Matheson, 2001][2].

**Examples**

(C.6) **T:** "...Gäbe es ein Element x $\in$ A $\cap$ B, dann wäre x $\in$ A und x $\notin$ K ( B ), was ein Widerspruch zur Annahme ist." (soc13k)
*[If there existed an element $x \in A \cap B$], then both $x \in A$ and $x \notin K$ ( $B$ ) would hold, which is a contradiction to the hypothesis.*

[2] In the DIALOG corpus, we had signs that students ignored the "computer" tutor, which would be specific to human-computer interaction [Tsovaltzi and Karagjosova, 2004]. Related research also exists in [Shechtman and Horowitz, 2003].

**C.3.2.1.2 Reassert** The same as *assert*, but the speaker believes that the claim has already been made and indicates this belief.

**Notes** Only statements that are old can be *reasserts*, since we take the speaker's intentions into account, not the objective familiarity status of the utterances.

*Reasserts* are probably informationally redundant utterances (IRUs), that is utterances that provide information that has already been established in the dialogue [Karagjosova, 2003].

**Obligations** Same as for *assert*, namely a *reassert* poses the obligation to address it. Any utterance addressing a *reassert* will discard this obligation.

**Examples**

(C.7) **T2:** "Sie müssen zuerst die wenn-dann-Beziehung auflösen" (`give-away-relevant-concept` hint) (soc5k)
*[First you have to eliminate the if-then relation]*
...
**T5:** "...Wir müssen ja zuerst die wenn-dann-Beziehung auflösen." (`elaborate-domain-object` hint)
*[... We have to eliminate the if-then relation first, though.]*

**C.3.2.1.3 Other-statement** Any *statement* that is neither an assert nor a reassert.

**Notes** This dialogue move is included for the sake of completeness.

### C.3.2.2 Influencing-addressee-future-action

The primary purpose of this aspect is to influence the addressee's future non-communicative actions as in the case of requests and suggestions. The hearer can coherently respond to it with "I can't do that". Questions also belong to this class if they suggest a course of action in addition to asking a question [Allen and Core, 1997]. This class is subdivided into *open-option* and *action-directive*.

**C.3.2.2.1 Open-option** An utterance that suggests a course of action or states a possibility [Allen and Core, 1997]. Imperatives are normally not *open-options*.

**Obligations** It poses an obligation to the student to address it. There are no obligation for the tutor.

**Examples**

(C.8) **T:** "Zuerst moechte ich, dass Sie…" (did15p)
*[First I want you to…]*

(C.9) **T:** "Sie muessen nun versuchen,…" (soc12d)
*[Now you have to try to…]*

(C.10) **T:** "Diesen Schritt muessen Sie noch naeher erlaeutern." (soc17k)
*[You have to clarify this step more precisely.]*

(C.11) **T:** "Vielleicht sollten Sie noch einmal in Ihren Begleitmaterial nachsehen." (soc20p)
*[Maybe you should have another look at your accompanying material.]*

(C.12) **S:** "Ich moechte die antwort wissen." (soc5k)
*[I want to know the answer.]* (*Resign* at task level)

**Counter-example**

(C.13) **S:** "nehmen wir an, dass…" (soc1k)
*[we assume that]*
(This is not an *open-option*, but a *statement*.)

**Relation to moves in other dimensions** *Open-option* may be a *hint* in the TASK dimension.

**C.3.2.2.2 Action-directive** An utterance that requests an action to be performed, i.e., commands, pleas, etc.

**Obligations** The listener is obliged to either perform the requested action or respond to the request, e.g., refuse to perform the action [Allen and Core, 1997]. In other words, she does not have the permission to ignore it [Traum and Allen, 1994]. The *action-directive* can also be rejected (e.g., the hearer refuses to perform the action, like the student saying "I want to try something else"), or it can be addressed (e.g., the hearer wants the speaker to clarify the request).

**Examples**

(C.14) **T1:** "Bitte zeigen Sie…" (soc5k)
*[Please show…]*

**Notes** The obligation is different from [Allen and Core, 1997] where *open-option* does not put an obligation on the hearer, i.e., the hearer can ignore it without any negative effect.

In the tutorial dialogue genre, both pose an obligation to address them. Hints can be realised either by an *open-option* or an *action-directive*, based on other issues, e.g., how strongly the tutor feels about the command being followed.

#### C.3.2.3 Info-request

An *info-request* is an utterance that requests information that the speaker does not possess.

**Obligations** In tutorial dialogues, the student is obliged to answer every tutor question. If the question is domain related, the student is obliged to resolve it or attempt to resolve it. This involves an obligation to answer, i.e., *assert*, *reject* or *ask* (*info-request*), which is a *request-clarification* at the BACKWARD-LOOKING function [Kreutel and Matheson, 1999]. This obligation does not hold for the tutor who can refuse to answer questions, or ignore them. The tutor will really only answer *info-requests* that are either non-domain related (they have no function at the task level), or otherwise based on teaching model considerations. Example C.15 illustrates the obligation. In **T4** the tutor performs an *info-request*. The student in **S3** ignores the tutor's question, so the tutor in **T5** states the obligation explicitly and not allowing the student to overlook it.

(C.15) **T4:** "Glauben Sie, daß Sie das nun beweisen müssen oder ist das eine Folgerung aus dem Obigen?" (soc23p)
*[Do you think that you have to prove this now, or does it follow from the previous?]*
**S3:** "$A \cap B \in P\ (\ (\ C \cup A\ ) \cap (\ C \cup B\ )\ ) = A \cap B \in P\ (\ C \cup A\ ) \cap P\ (\ C \cup B\ )$" (unknown *domain-contribution*)
**T5:** "Bitte beantworten Sie zuerst meine Frage, bevor Sie mit dem Beweis fortfahren."
*[Please, answer my question first, before you move on with the proof.]*

**Notes** An *info-request* performed by the tutor initiates a clarification subdialogue.

**Examples**

(C.16) **T:** "..., aber wie bringt das den Beweis weiter?" (soc1p)
*[..., but how does this help with the proof?]* (This utterance is an *info-request* and not a *diagnostic-query*, because the tutor really does not know the answer.)

(C.17) **T:** "..., aber was hat die Potenzmenge mit diesem Beweis zu tun?" (soc1k)
*[..., but what does the powerset have to do with this proof?]* (This is indeed a *diagnostic-query*. The tutor knows that the powerset has nothing to do with this proof. It is a rhetorical question, a kind of prompting students to correct their own mistake.)

**Relation to other moves in the same dimension** In DAMSL, an *info-request* is any utterance that creates an obligation for the hearer to provide information. It includes all questions: yes/no-questions, WHO-questions, assertive questions, indirect questions (e.g., "Tell me the time"), requests for other actions that provide information (e.g., "Show me where the city is on the map").[3] However, since we also consider *diagnostic-queries*, we only treat the latter as an *info-request* (See C.3.2.4).

It is different from *understanding-query* that just asks the collocutor to signal that she understands.

**Relation to moves in other dimensions** It is different from *request-clarification*, which requests specific information about a previous vague utterance.

#### C.3.2.4 Diagnostic-query

An utterance with which "the speaker is testing whether the listener knows a piece of information by asking him to supply the information". The speaker, normally the tutor, already knows the answer. It is often a question, but can be also a request, e.g., "...tell me how electricity flows through the circuit" [Core *et al.*, 2002].

**Obligations** It poses the obligation on the student to answer it (addressing it is not enough here).

**Examples**

(C.18) **T:** "(Sie haben die Regel aber nicht richtig angewendet.) Wie haetten Sie diese Regel anwenden muessen?" (soc1p)
*[(Only you did not apply the rule correctly). How should you have applied the rule?]*

**Counter-example**

(C.19) **T:** "Können Sie das noch genauer erklaeren?" (soc20k)
*[Can you explain that more precisely?]*
(This is not a *diagnostic-query*, because the tutor does not know what to expect as an answer. It is a *check-origin-problem* at the task level, realised as a request-clarification and *into-request* or *open-option*).

[3] We would rather consider those as *action-directives*.

**Relation to other moves in the same dimension** *Diagnostic-query* differs from *info-request* and *understanding-query* in that the speaker already knows the answer.

**Relation to moves in other dimensions** It can be a *hint*, an *align*, or a *checkorigin-problem* in the TASK dimension.

#### C.3.2.5 Understanding-query

An utterance that asks the listener whether she understood without making her prove it [Core *et al.*, 2002]. It takes a "yes" or "no" as an answer.

**Obligations** It poses the obligation to address it. It does not pose the obligation to explain more or to prove it.

**Examples**

(C.20) **T:** "Verstehen Sie jetzt, wie das mit der Implikation zusammenhaengt?" (soc12k)
*[Do you understand now how this is connected to the implication?]*

**Relation to other moves in the same dimension** It is different from *diagnostic-query* where the speaker does not know the answer. The realisation of the two can be the same (e.g., "Do you understand?"), but the obligations that they pose are different; *diagnostic-queries* require more than "yes" or "no" as an answer.

It is different from *info-request*, because it asks the speaker to signal understanding.

#### C.3.2.6 Committing-speaker-future-action

Utterances that potentially commit the speaker to some future course of action. The commitment can be conditional on the listener's agreement (*offer*) or not (*commit*) [Allen and Core, 1997].

**C.3.2.6.1 Commit** An utterance with which the speaker commits herself to a future course of action.

**Obligations** *Commit* does not pose an obligation on the hearer to respond, but poses a future obligation on the speaker to perform the action she has committed herself to.

**Examples**

(C.21) **T4:** "Dies mache ich nun." (did16k)
*[I'm going to do this now.]*
(This is a *discourse-marking* at the communication management level.)

**Notes** An utterance that accepts an *action-directive* or *open-option* will typically be a *commit* [Allen and Core, 1997]. The speaker's commitment does not depend on the acceptance of the commitment by the hearer, e.g., as in the case of a promise.

**C.3.2.6.2 Offer** An utterance by which the speaker indicates willingness to commit to an action if the hearer accepts it [Allen and Core, 1997].

**Obligations** In the tutorial genre, it does not pose an obligation on students to address it explicitly, because they are obliged to accept it. In the same way, the tutor is not obliged to wait for the student to accept it in order to perform the offered action.

If the student makes an *offer*, the tutor is obliged to address it.

**Examples**

(C.22) **T:** "Let's look at an example" (constructed)

In [Allen and Core, 1997], *offer* poses an obligation on the hearer to address it (accept or reject also count as addressing), and on the speaker to bring about the action that she committed to.

### C.3.2.7 Conventional

This category captures the function of utterances as conventional communicative actions, such as greetings and saying goodbye. In [Allen and Core, 1997] conventional includes also explicit performatives (e.g., "You are fired"), exclamations (e.g., "Ouch"), and FORWARD-LOOKING functions not captured by the scheme, such as holding/grabbing the turn (e.g., "Right", "Okay").

**C.3.2.7.1 Conventional-opening** An utterance that is a phrase conventionally used to summon the addressee and/or start the interaction (e.g., "Can I help you", "Hi") [Allen and Core, 1997].

**Obligations** The collocutor is obliged to address it.

**C.3.2.7.2 Conventional-closing** An utterance that is a phrase conventionally used in a dialogue closing or used to dismiss the addressee (e.g "Goodbye") [Allen and Core, 1997].

**Obligations** None.

#### C.3.2.8 Apologise

An utterance by which the speaker expresses regret. According to [Searle, 1975], it is an expressive speech act, that is, the speaker expresses a psychological state or reaction.

**Obligations** None.

**Examples**

(C.23) **S:** "(Hab keine ahnung mehr.) Tut mir leid!" (min14k)
*[I have no idea any more. I'm sorry!]*

(C.24) **T:** "Das ist nicht richtig."
*[That is not right.]*
**S:** "Das tut mir leid." (constructed)
*[I'm sorry]*
(The student performs an *address-statement*, BACKWARD-LOOKING function, and an *apologise*, FORWARD-LOOKING function.)

(C.25) **S:** "Entschuldigung, es gilt natuerlich..." (soc20p)
*[Excuse me, of course it holds...]*
(This is not a *self-correction*, because the tutor has pointed the student to the material. Otherwise, every student move after a hint would have to be a *self-correction*. Instead, we consider "Entschuldigung" an *apologise*, and what follows it *asserts*.)

**Relation to moves in other dimensions** Its BACKWARD-LOOKING function is *address-statement*.

**Notes** It is FORWARD-LOOKING, because it does not arise from a previous obligation/intention.

There is a move *apology* in the DARPA 9 Communicator system [Walker and Passonneau, URL] that captures utterances that the system produces to apologise for misunderstandings, e.g., "I am sorry. I am having trouble understanding you."

#### C.3.2.9 Gratitude

An utterance that expresses gratitude.

**Obligations** There is no obligation to address it, but the listener may say "You're welcome".

**Examples**

(C.26) **S:** "Danke." (soc20k)
*[Thank you]*
(This was a reaction to an *encourage*)

(C.27) **S:** "That is nice of you." (constructed)
("Thank you" cannot be an *assert*. "That is nice of you." can.)

**Notes** It is FORWARD-LOOKING, because it does not arise from a previous obligation/intention.

**Relation to moves in other dimensions** Its BACKWARD-LOOKING function is *address-statement*.

#### C.3.2.10 Signalling-emotion

We use this, instead of *exclamation* such as "Ouch", which is used in [Allen and Core, 1997].

**C.3.2.10.1 Frustration** An utterance that expresses a negative emotion.

**Obligations** The tutor is obliged to do an *encourage*. This is captured in the TASK dimension, where such an utterance is considered a *resign*.

**Examples**

(C.28) **S:** "Wenn ich das wuesste!" (soc20k)
*[If only I knew this!]*

**Relation to moves in other dimensions** It can realise a *resign* move at TASK level.

**C.3.2.10.2 Satisfaction** An utterance that expresses a positive emotion.

**Obligations** None.

**Examples**

(C.29) **S:** "I did well!" (constructed)

#### C.3.2.11 Other-forward-function

This captures any action not captured by any other FORWARD-LOOKING function [Allen and Core, 1997].

### C.3.3 Backward-Looking Dimension

This dimension captures how the current utterance relates to the previous discourse. Only direct (explicit) speech acts are considered.

1. Agreement
   (a) Accept
   (b) Accept-part
   (c) Maybe
   (d) Reject
   (e) Reject-part
2. Understanding
   (a) Signal-non-understanding (SNU)
   (b) Request-clarification
   (c) Signal understanding
      i. Repeat-rephrase
      ii. Acknowledge
      iii. Completion
   (d) Correct-misspeaking
3. Answer
4. Information-relation
   (a) Address-action-dir
   (b) Address-question
   (c) Address-statement
   (d) Address-SNU
   (e) Address-other

#### C.3.3.1 Agreement

This aspect covers how the utterance unit affects what the participants believe they have agreed to, typically at the TASK level, or whatever the topic of discussion is. These relations occur in contexts where the one agent has made some kind of proposal, such as a request that the hearer does something, an offer that the speaker does something, or a claim about the world. The current utterance then indicates the other participant's view of the proposal. An agent may explicitly accept or reject all or part of the proposal, be non-committal, or leave it open by requesting additional information, or exploring the consequences.

For instance, the utterance in Example C.30 is not explicitly accepting the student utterance.

(C.30) **T:** Sehr gut! (soc20k)
*[Very well!]*

(C.31) **T:** "There's something missing." (constructed)
(This is accept at TASK level, because it accepts what was said and indicates additionally that something is missing.)

An utterance may explicitly accept part of the previous utterance, but implicitly reject part of it, however, this aspect only covers what is explicitly accepted or rejected by a response [Allen and Core, 1997].

**C.3.3.1.1 Accept** An utterance that accepts a proposal, request, statement or information request. The following is an example from [Allen and Core, 1997]:

(C.32) **A:** "Can you tell me the time?"
**B:** "Yes" (*accept*)

**Obligations** Whenever a FORWARD-LOOKING move is accepted, there is the obligation to carry the accepted move through [Matheson *et al.*, 2000]. For instance, an accepted offer incurs an obligation to follow it. An acceptance of a suggestion or request to perform an action poses the obligation to attempt to achieve the action [Traum and Allen, 1994].

*Accept*, however, does not apply to all FORWARD-LOOKING moves. Moreover, the acceptance of an *assert* does not pose any obligations to perform any actions.

**Examples**

(C.33) **T:** "Der Ansatz ist richtig." (soc2k)
*[The approach is correct]*

**C.3.3.1.2 Accept-part** An utterance that accepts part of a proposal, request, statement or information request. Implicitly, it rejects another part of the utterance, but this only covers what is explicitly accepted. When an utterance does both explicitly, it consists of two segments.

**Obligations** Same as for *accept* for the part that is accepted.

**Examples**

(C.34) **T:** "P ( C ) $\cup$ P( A $\cap$ B ) $\subseteq$ P ( C $\cup$ ( A $\cap$ B )): das ist richtig! A $\cap$ B $\subseteq$ P ( A $\cap$ B ): das ist nicht richtig!" (min11p)
*P ( C ) $\cup$ P( A $\cap$ B ) $\subseteq$ P ( C $\cup$ ( A $\cap$ B )): this is right! A $\cap$ B $\subseteq$ P ( A $\cap$ B ): this is not right!]*

**C.3.3.1.3 Maybe** An utterance with which the speaker "explicitly states that he cannot give a definite answer at the moment" [Allen and Core, 1997].

**Obligations** None.

**Examples**

(C.35) "I'll have to think about it." (constructed)

**Notes** It is a response to an *offer*.

In our domain it is not likely to have a *maybe*, because it is clear that one has to think before responding.

**C.3.3.1.4 Reject** An utterance that rejects a proposal, request, statement or information request. It says nothing positive about the antecedent and possibly indicates an error. It can be implicit or contain "no" or "not" [Core *et al.*, 2002].

**Obligations** Student are obliged to justify or elaborate on a *reject* that they perform.

**Examples**

(C.36) **T:** "Das ist keine Aussage" (*signal-ill-formed*) (did18d)
*[This is not an expression]*

(C.37) **T:** "Das kann man nicht so folgern" (soc13k)
*[One cannot infer this]*

(C.38) **T:** "Das ist nicht der richtige Weg" (soc1k)
*[That is not the right way]*

(C.39) **T:** "Was Sie geantwortet haben, ist nicht eindeutig." (soc12p)
*[What you replied is not unambiguous.]*

(C.40) **T:** "Das ist keine vollstaendige Aussage" (did15p)
*[This is not a complete utterance]*

(C.41) **T:** "Das ist kein vollstaendiger Ausdruck." (did19p)
*[This is not a complete expression.]*

(C.42) **T:** "Sie haben die Regel aber nicht richtig angewendet." (soc1p)
*[Only you have not applied the rule correctly]*

**Note:** In tutorial dialogues, "no" is very seldom used, and only when the student is totally wrong.

**C.3.3.1.5 Reject-part** An utterance that rejects partly a proposal, request, statement or information request. It implicitly accepts another part of the utterance, but *reject-part* only covers what is explicitly rejected.

**Obligations** As in *reject*, the tutor is obliged to justify the rejection or elaborate on it.

The tutor might signal which part is being rejected. This depends on the teaching model and on the possibility of realising a *domain-contribution-evaluation*, which indicates the problematic part.

**Examples**

(C.43) **T:** "Das ist nicht ganz richtig" (did4p)
*[This is not totally right]*

(C.44) **T:** "P ( C ) $\cup$ P( A $\cap$ B ) $\subseteq$ P ( C $\cup$ ( A $\cap$ B )): das ist richtig! A $\cap$ B $\subseteq$ P ( A $\cap$ B ): das ist nicht richtig!" (min11p)
*P ( C ) $\cup$ P( A $\cap$ B ) $\subseteq$ P ( C $\cup$ ( A $\cap$ B )): this is right! A $\cap$ B $\subseteq$ P ( A $\cap$ B ): this is not right!]*

### C.3.3.2 Understanding

**C.3.3.2.1 Signal-non-understanding (SNU)** An utterance that signals that the speaker has not understood the previous utterance, i.e., did not hear it or could not make sense of it.

Instances for that move are "I don't understand" and variants of it like "What did you say?".

**Obligations** *SNU* poses an obligation to address it (e.g., via a clarification).

**Examples**

(C.45) **T** "Ich verstehe Ihre Frage nicht" (did15k)
*[I don't understand your question]*

(C.46) **T** "Was wollen Sie damit sagen?" (soc17k)
*[What do you mean by that?]*

(C.47) **T** "Was meinen Sie?" (soc17d)
*[What do you mean?]*

**Relation to other moves in the same dimension** The difference to *request-clarification* is the following: An utterance is an *SNU*, when the interlocutor does not understand anything of the preceding contribution, and a *request-clarification* when she understands, but not completely [Core *et al.*, 2002].

**Relation to moves in other dimensions** An *SNU* can have the FORWARD-LOOKING functions *statement* or *info-request*.

**Notes** *SMUs* are "utterances that explicitly indicate a problem in understanding the antecedent". An applicability test is the rough paraphrase "What did you say/mean?" [Allen and Core, 1997]. We have narrowed down this definition to cases like "I don't understand" and variants. [Allen and Core, 1997] point out that not all clarification questions are also *SNUs*, they could be *holds*, for example.[4]

An *SNU* can be addressed (*address-SNU*), when it is a question, and it can also be responded to. The response is then an *answer*, or if the *SNU* is not resolved, an *address-SNU*.

*SNUs* introduce clarification subdialogues.

For NL realisation purposes we distinguish between *SNU* and *request-clarification*. The formulation of one or the other can prove useful for letting the collocutor know how much she needs to clarify.

**C.3.3.2.2 Request-clarification** A move that applies when some of the input has been understood [Core *et al.*, 2002]. It has, basically, the form of a question, but can be also an imperative, i.e., it can be an *action-directive* in the FORWARD-LOOKING dimension.

**Obligations** It poses an obligation to address it (e.g., via a clarification).

**Examples**

(C.48) **T:** "Was meinen Sie mit…" (did15p)
*[What do you mean by…]*

(C.49) **T:** "Meinten Sie wirklich…?" (did16p)
*[Did you really mean…]*

(C.50) **T:** "Was soll das x darstellen?" (soc17p)
*[What is x supposed to represent?]*

(C.51) **T:** "Was soll das heissen?" (did19k)
*[What is that supposed to mean?]*

(C.52) **T:** "Ist das die antwort auf meine Frage oder ein neuer Loesungsversuch?" (soc21p)
*[Is this the answer to my question or an attempt at a new proof?]*

(C.53) **T:** "Bitte erklaeren Sie Ihren Schritt genauer!" (soc12p)
*[Please, explain your step more precisely!]*

**Relation to moves in the same dimension** *SNU* applies when none of the input has been understood.

[4] *Hold* in DAMSL is meant as a reaction to proposals. It seems to be an attempt to have an all-encompassing move, instead of accounting for subdialogues. This is not sufficient for us, as subdialogues are important in tutorial dialogues.

**Notes** There is no move *request-clarification* in DAMSL.
*Request-clarifications* are only questions in [Core *et al.*, 2002].

**C.3.3.2.3 Signal understanding** It is an utterance that signals understanding. Any utterance that does not explicitly signal non-understanding implicitly indicates understanding, so we don't cover this [Allen and Core, 1997]. It comprises three subclasses.

- **Repeat-rephrase** This move is "...used for utterances that repeat or paraphrase what was just said in order to signal that the speaker has been understood... [*repeat-rephrases*] do not necessarily make any further commitment as to whether the responder agrees with or believes the antecedent." [Allen and Core, 1997]

  **Obligations** None.

  **Notes** It is in nature really an *acknowledge*.

- **Acknowledge**

  "Acknowledgements are utterances consisting of short phrases such as *OK, yes, uh-huh*, that signal that the previous utterance was understood without necessarily signalling acceptance" [Allen and Core, 1997]. They do not resolve the content of the utterance that they address.

  **Obligations** When the student addresses the tutor, the tutor is obliged to *acknowledge*, even if it is only to reject the offer to talk about the issue that the student has raised. The preferred way of doing the latter is in an explicit manner that would be too condescending in other genres. For example, "Before we get to that,...", is implicitly acknowledges the previous utterance, but is actually *reject*. This is the case, because the tutor does not give direct answers, and therefore she must indicate somehow that she is taking the student's answer into account, so as not to give the impression that she is ignoring it. However, students are not obliged to perform any *acknowledge*. Sometimes they don't do any implicit acknowledging either. They are just silent and assume that the tutor knows that they are listening and thinking about the problem. Otherwise, the student would ask for clarification. An explanation for that is that both student and tutor are aware of the obligations that their respective social roles carry. Since it is the students' obligation to take what the tutor says into account in order to proceed with the task (expertise plays a role), students do not feel that they need to indicate that this is what they are doing [Tsovaltzi and Matheson, 2002].

**Examples**

(C.54) **S:** "schon klar." (soc20k)
*[that is clear.]*

**Relation to other moves in the same dimension** It might well be the case that "yes", "OK" and "right" are *accepts*. In that case the tutor allows the task to move on and does not insist on eliciting the correct domain contribution.

"Good", "very good", "that's right" serve always as *accept*. Whereas "right", "all right" and "OK" are ambiguous. They are commonly only an *acknowledge* when they are followed by a *hint*. "OK" is often used after a not totally wrong answer.

**Notes** There is only a realisation difference from *repeat-rephrase* and *acknowledge*. Namely, that *acknowledge* can only be a short response.

When the tutor does a *check-origin-problem*, there is normally no *acknowledge* preceding it.

- **Completion**

  An utterance that shows "understanding by finishing or adding to the clause that a speaker is in the middle of constructing" [Allen and Core, 1997].

  **Obligations** None.

**C.3.3.2.4 Correct-misspeaking** They are "... utterances that by offering a correction indicate that the hearer believes that the speaker has not said what she actually intended" [Allen and Core, 1997]. It applies only to cases where the current speaker makes a correction to what was previously uttered, that is, to the utterance that is addressed by the current correct-misspeaking.

There is no dimension for annotating self-corrections currently in DAMSL.

**Obligations** None.

**Examples**

(C.55) **S:** "A is a subcategory of B." (where "subcategory" is the wrong terminology) (constructed)
**T:** "That's right, A is a sub*set* of B." (*correct-misspeaking*) "But why?"

**Relation to moves in other dimensions** It can be a `correct-info` hint in the TASK dimension.

#### C.3.3.3 Answer

An utterance that complies with an info-request action in the antecedent and resolves it.[5]

**Obligations** None.

**Examples**

(C.56) **T:** "Ist das noch derselbe Lösungsweg wie in der vorigen antwort?" (*Info-request*) (did10p)
*[Is this the same solution as in the previous answer?]*
**S:** "Nein" (correct *domain-contribution*)
*[No]*
"ich habe mich umentschieden: Ich zerlege jetzt die Potenzmenge: $P ( C \cup ( a \cap B ) ) \supseteq P ( C ) \cup P ( a \cap B )$" (*answer* to **T**)
*[I changed my mind. I am now using the powerset: $P ( C \cup ( a \cap B ) ) \supseteq P ( C ) \cup P ( a \cap B )$]*

**Relation to moves in other dimensions** *Answers* will always be *asserts* in the FORWARD-LOOKING dimension even if they are imperative, since the FORWARD-LOOKING function is to provide information, not to influence the future action [Allen and Core, 1997].

#### C.3.3.4 Information-relation

This category "captures how the content of the current utterance relates to the content of its antecedent". It is not further elaborated in DAMSL, but is left for future study [Allen and Core, 1997]. It is useful in connection to the proof manager and in disambiguating between what a current utterance is and what its relation to the overall context is, e.g., in case a subdialogue intervenes. As pointed out before, *address-sth* applies when neither *accept*, *reject* or *answer* apply. These entail an implicit *address* anyhow.

The following categories are all new, defined based on the DIALOG corpus.

**C.3.3.4.1 Address-action-directive** Any utterance that addresses a previous (not necessarily an immediately preceding turn) *action-directive* move, but is not a *reject*, *reject-part*, *accept*, or *accept-part*.

**Obligations** None.

[5] See footnote on aboutness vs. resolvedness.

**Examples**

(C.57) **T:** "Bitte zeigen Sie: K ( ( a ∪ B ) ∩ ( C ∪ D ) ) = ( K ( a ) ∩ K ( B ) ) ∪ ( K ( C ) ∩ K ( D ) )!" (did10d)
*[Please show: K ( ( a ∪ B ) ∩ ( C ∪ D ) ) = ( K ( a ) ∩ K ( B ) ) ∪ ( K ( C ) ∩ K ( D ) )!]* (*action-directive*)
**S:** "K ( ( a ∪ B ) ∩ ( C ∪ D ) ) = K ( a ∪ B ) ∪ K ( C ∪ D )"
(a correct *domain-contribution* and *address-action-directive* to **T**)

**C.3.3.4.2 Address-question** Any utterance that addresses a preceding *info-request*, *understanding-query*, or *diagnostic-query* move without resolving it.

**Obligations** None.

**Examples**

(C.58) **S:** "was ist K ( a )" (*info-request*) (min11d)
*[What is K ( a )]*
**T:** "Das kann ich nicht beantworten." (*address-info-request* to **S**)
*[I cannot answer this.]*

**C.3.3.4.3 Address-statement** Any utterance that addresses a preceding *statement*, *assert* or *reassert*, without being an explicit *reject*, *reject-part*, *accept*, or *accept-part*.

**Obligations** None.

**Examples**

(C.59) **S:** "das stimmt schon. verstehe die definition nicht, einfaches Beispiel wuerde mir weiter helfen" (*statement*) (did15p)
*[that is of course correct. I don't understand the definition, a simple example would help me more]*
**T:** "Sei die Menge X=$\{1, 2\}$. Dann ist die Potenzmenge von X die Menge P(X)= $\{0, \{1\}, \{2\}, \{1, 2\}\}$." (*address-statement* to **S**)
*[Let the set X=$\{1, 2\}$. Then the powerset of X is the set P(X)= $\{0, \{1\}, \{2\}, \{1, 2\}\}$.]*

**Relation to moves in other dimensions** In the TASK dimension, such questions can be pragmatic hints, e.g., *elicit-discrepancy*, or a *check-origin-problem*.

**Notes** We use this instead of *followup* that is a BACKWARD-LOOKING move in BE&E, e.g., "Why do you think that?", "How are you going to do that?". *Followup* is defined as a reaction of the tutor to a student answer in form of a followup question asking for more detail, or asking a question about an answer [Core *et al.*, 2002]. An example from our corpus is "Warum?" *[Why?]*, (soc13k).

**C.3.3.4.4 Address-SNU** Any utterance that addresses a previous *SNU*.

**Obligations** None.

**Examples**

(C.60) **T:** “Ich verstehe Ihre antwort nicht, denn das ist kein vollständiger deutscher Satz: Den Durchschnitt der Menge K ( a ) und der Menge K ( B ), also K ( ( a ∪ B ) ∩ ( C ∪ D ) )” (soc21d)
*[I don’t understand your answer, because it is not a complete German sentence: Then union of the set K ( a ) and of the set K ( B ), therefore K ( ( a ∪ B ) ∩ ( C ∪ D ) )](SNU)*

**S:** “Ist x der Durchschnitt der Menge K ( a ∪ B ) und der Menge K ( C ∪ D ), dann ist K ( ( a ∪ B ) ∩ ( C ∪ D ) )” (a wrong *domain-contribution* and an *address-SNU* to **T**)
*[If x is the union of the set K ( a ∪ B ) and of the set K ( C ∪ D ), then it is K ( ( a ∪ B ) ∩ ( C ∪ D ) )]*

**Obligations** None.

**C.3.3.4.5 Address-other** Utterances addressing other moves apart from the ones defined above and apart from *reject*, *reject-part*, *accept*, or *accept-part*, which also address utterances. e.g. *address-request-clarification*, *address-diagnostic-query*, *address-understanding-query*, *address-open-option*, *address-commit*, *address-offer*, *address-emotion*, *address-apologise*, *address-gratitude*, *address-agreement*, *address-signal-understanding*, *address-correct-misspeaking*, *address-answer*.

For the analysis of the moves in the dimensions TASK, CONVENTIONAL-TASK MANAGEMENT, and CONVENTIONAL-COMMUNICATION MANAGEMENT see Chapter 4.

## C.3.4 Communicative Status Dimension

This dimension captures features of an utterance unit such as whether it was interpretable. These features mark exceptional cases, therefore, most utterance units won’t have functions in this dimension [Allen and Core, 1997]. The features are:

- Uninterpretable
- Abandoned
- Self-talk

### C.3.4.1 Uninterpretable

Utterances that are not comprehensible fall under this category. These utterances are usually word fragments, or utterances containing misspelled or mispronounced words such that it is impossible to understand them.

#### C.3.4.2 Abandoned

An utterance or utterance fragment that does not provide content to the dialogue, i.e., "the import of the dialogue would not change if these utterance units were removed" [Allen and Core, 1997].

**Notes** In written communication, such utterances are likely not to be submitted, i.e., to be deleted before submission.

#### C.3.4.3 Self-talk

"The utterance unit consists of one speaker talking to himself." [Allen and Core, 1997]. It does not normally occur in written communication.

# Appendix D

# NL Examples of Hint Categories

In this appendix we present a more detailed list of the NL examples of the hint categories in Chapter 4 whose specifications include the Domain Technique. Therefore, we follow a listing based on the Domain Technique.

`Give-away-inference-rule`

### NL Examples

NL Template: "You have to use $<$ Rule of Inference $>$ of $<$ Relevant Concept $>$."

- Occurrence state of definitions or substitutions: "You have to use the definition of powerset."
- Case distinction: "You have to use the cases deriving from the disjunctive definition of *the union* $\cup$.", where *the union* $\cup$ is the disjunctively defined concept, defined as $U \cup V = \{x | x \in U \text{ or } x \in V\}$ and the cases deriving from its definition would be $x \in U$ or $x \in V$.
- Induction: "You have to use the steps deriving from the inductive definition of *the set of all finite subsets of* $X$.", where *the set of all finite subsets of* $X$ is the inductively defined concept, and $P_f(X)$ *is the set of all finite subsets of* $X$ *if: (i)* $\emptyset \in P_f(X)$ *(ii)* $A \in P_f(X), x \in X \Rightarrow A \cup \{x\} \in P_f(X)$). For example, if we want to prove that $\Phi(B)$ holds for all $B \in P_f(X)$, the steps are (i) $\Phi(\emptyset)$ and (ii) for $A \in P_f(X)$ and $x \in X$ if $\Phi(A)$ holds, then $\Phi(A \cup \{x\})$ holds.
- Occurrence state of quantifiers (where the Relevant Concept is the quantifier type):
  - Universal quantifier: "You have to use the elimination of the *for all*."

- Existential quantifier: "You have to use the elimination of the *there exists*."

- Occurrence state of connectives (where the Relevant Concept is the connective type):
  - Equivalence: "You have to use the elimination of the $\Leftrightarrow$."
  - Equality: "You have to use the elimination of the equality =."
  - Implication: "You have to use the elimination of the implication $\Rightarrow$."
  - Conjunction: "You have to use the elimination of the conjunction *and*."
  - Disjunction: "You have to use the elimination of disjunction *or* for the disjunction."

`Give-away-basic-knowledge`

**NL Examples**

NL Template: "The rule for the < Basic Knowledge Reference > is: < Basic Knowledge >."

- Occurrence state of definitions or substitutions: "The rule for the definition of powerset is: $P(V) = \{U|U \subseteq V\}$."
- Case distinction: "The rule for the case distinction of the *union* $\cup$ is: for the expression to hold for the union of $U$ and $V$ it has to hold for the different cases deriving from its disjunctive definition $x \in U$ or $x \in V$."
- Induction: "The rule for the induction of *the set of all finite subsets of* $X$ is: for the expression to hold for *the set of all finite subsets of* $X$, it has to hold for the different steps deriving from its inductive definition $\emptyset \in P_f(X)$ and $A \in P_f(X), x \in X \Rightarrow A \cup \{x\} \in P_f(X)$.
- Occurrence state of quantifiers:
  - Universal quantifier: "The rule for the elimination of the *for all* is: for an expression to hold for all x, it has to hold for an arbitrary but fixed x."
  - Existential quantifier: "The rule for the elimination of the *there exists* is: if there exists an x for which something holds, then it has to hold for at least one particular x."
- Occurrence state of connectives:
  - Equivalence: "The rule for the elimination of equivalence $\Leftrightarrow$ is: an expression with an equivalence holds if it holds from left to right and from right to left".

– Equality: "The rule for the elimination of equality = is: an expression with an equality holds if you assume the left hand-side and you can prove the right hand-side."[1]
– Implication: "The rule for the elimination of implication is: an expression with an implication $\Rightarrow$ holds if you assume the hypothesis and you can prove from that the conclusion."
– Conjunction: "The rule for the elimination of conjunction *and* is: an expression with a conjunction holds if it holds for every term connected by the *and* separately."
– Disjunction: "The rule for the elimination of disjunction *or* is: an expression with a disjunction holds if it holds for at least one of the terms connected by the *or*."

`Elicit-subordinate-concept-meta-reasoning`

**NL Examples**

NL Template: "Think of what you need to prove and how you can connect that to the < Relevant Concept >."

- Occurrence state of definitions or substitutions: "Think of what you need to prove and how you can connect that to the powerset."
- Case distinction: "Think of what you need to prove and how you can connect that to the *union* $\cup$.", where the < Relevant Concept > is the disjunctively defined concept, here the *union* $\cup$.
- Induction: "Think of what you need to prove and how you can connect that to *the set of all finite subsets of X*.", where the < Relevant Concept > is the inductively defined concept, here *the set of all finite subsets of X*.
- Occurrence state of quantifiers (where the < Relevant Concept > is the quantifier type):
  – Universal quantifier: "Think of what you need to prove and how you can connect that to the *for all*."
  – Existential quantifier: "Think of what you need to prove and how you can connect that to the *there exists*."
- Occurrence state of connectives (where the < Relevant Concept > is the connective type):
  – Equivalence: "Think of what you need to prove and how you can connect that to the equivalence $\Leftrightarrow$."
  – Equality: "Think of what you need to prove and how you can connect that to the equality =."

[1] Note that this is a definition of equality only for set theory.

– Implication: "Think of what you need to prove and how you can connect that to the implication $\Rightarrow$."
– Conjunction: "Think of what you need to prove and how you can connect that to the conjunction *and*."
– Disjunction: "Think of what you need to prove and how you can connect that to the disjunction *or*."

`Give-away-subordinate-concept-meta-reasoning`

**NL Examples**

NL Template: "You can consider the < Subordinate Concept > and how it connects to the < Relevant Concept >."

- Occurrence state of definitions or substitutions: "You can consider the subset and how it connects to the powerset."
- Case distinction: "You can consider the different cases that you have to prove deriving from the inductive definition of the least set.", where the < Relevant Concept > is the disjunctively defined concept, here the *union* $\cup$.
- Induction: "You can consider the different steps that you have to prove, deriving from the inductive definition of *the set of all finite subsets of X*.", where the < Relevant Concept > is the inductively defined concept, here *the set of all finite subsets of X*.
- Occurrence state of quantifiers:
  NL Template: "In order for the expression to hold for < Relevant Concept > we need to prove it for < Subordinate Concept >", where the Relevant Concept is the quantifier type and the Subordinate Concept is the new goal formula.
  – Universal quantifier: "In order for the expression to hold for the *for all* we need to prove it for some arbitrary x, y, ..."
  – Existential quantifier: "In order for the expression to hold for some term, we need to prove it for a specific term <T>."
- Occurrence state of connectives:
  NL Template: "In order for the expression with the < Relevant Concept > to hold, you have to prove < Subordinate Concept >", where the Relevant Concept is the connective type and the Subordinate Concept the new goal formula.
  – Equivalence: "In order for the expression with the equivalence $\Leftrightarrow$ to hold, you have to prove both directions of the expression"
  – Equality: "In order for the expression with the equality = to hold, you have to prove the right hand-side of it."

– Implication: “In order for the expression with the implication $\Rightarrow$ to hold, you have to assume the left hand-side of it and from that prove the right hand-side.”

– Conjunction: “In order for the conjunction *and* to hold, you have to prove all terms connected by it.”

– Disjunction: “In order for the disjunction *or* to hold, you have to prove at least one of the terms connected by it.”

**Notes** The instantiation of the subordinate concept can be derived from the new goal formula.

`Elicit-domain-technique`

**NL Examples**

- Occurrence state of definitions or substitutions: “What you should you do here in order to deal with the $<$ Relevant Concept $>$, e.g., the powerset, or *if-then* relation, etc.”

- Case distinction: “What should you do here in order to deal with the disjunctive definition of the $<$ Relevant Concept $>$ ?”, e.g., *union* (i.e., the disjunctively defined concept).

- Induction: “What should you do here in order to deal with the inductive definition of $<$ Relevant Concept $>$ ?”, e.g., *the set of all finite subsets of* $X$ (i.e., the inductively defined concept).

- Occurrence state of quantifiers: “What should you do here in order to deal with the $<$ Relevant Concept $>$ ”, e.g., the *for all* (i.e., the quantifier type, which can be *for all* or *there exists*).

- Occurrence state of connectives: “What should you do here in order to deal with the $<$ Relevant Concept $>$ ?”, e.g., the equivalence $\Leftrightarrow$ (i.e., the connective type, which can be any of equivalence, equality, implication, conjunction, or disjunction).

`Give-away-domain-technique`

**NL Examples**

NL Template: “You have to $<$ Domain Technique $>$ the $<$ Relevant Concept $>$.”

- Occurrence state of definition: “You have to get rid of the powerset.”, where the $<$ Domain Technique $>$ is extract.

- Case distinction: “You have to apply case distinction to the *union* $\cup$”, where the Relevant Concept is the disjunctively defined concept, here the *union* $\cup$.

- Induction: "You have to apply induction of *the set of all finite subsets of* $X$", where the $<$ Relevant Concept $>$ is the inductively defined concept, here *the set of all finite subsets of* $X$.

- Occurrence state of quantifier: "You have to get rid of the *for all.*", where the $<$ Domain Technique $>$ is extract and $<$ Relevant Concept $>$ is the quantifier type, which can be *for all*, or *there exists*.

- Occurrence state of connective: "You have to get rid of the equivalence $\Leftrightarrow$.", where $<$ Domain Technique $>$ is extract and the $<$ Relevant Concept $>$ is the quantifier type, which cab be equivalence, equality, implication, conjunction, or disjunction.

`Elicit-connect-relevant-subordinate-concept`

**NL Examples**

NL Template: "Think of a theorem or lemma that you can apply and involves the $<$ Relevant Concept $>$ and the $<$ Subordinate Concept $>$."

- Occurrence state of definitions or substitutions: "Think of a theorem or lemma that you can apply and involves the powerset and the subset."

- Case distinction: "Think of a rule that you can apply and involves the disjunctively defined concept *union* $\cup$ and what you have to prove."

- Induction: "Think of a rule that you can apply and involves the inductively defined concept *set of all finite subsets of* $X$ and what you have to prove."

- Occurrence state of quantifiers (where the $<$ Relevant Concept $>$ is the Quantifier Type and $<$ Subordinate concept $>$ is the new goal formula.):

  – Universal: "Think of a rule that you can apply and involves the *for all* and what you need to prove."
  – Existential: "Think of a rule that you can apply and involves the *there exists* and what you need to prove."

- Occurrence state of connectives (where the $<$ Relevant Concept $>$ is the connective type and $<$ Subordinate Concept $>$ is the new goal formula.):

  – Equivalence: "Think of a rule that you can apply and involves the equivalence $\Leftrightarrow$ and what you need to prove."
  – Equality: "Think of a rule that you can apply and involves the equality $=$ and what you need to prove."
  – Implication: "Think of a rule that you can apply and involves the implication and what you need to prove."
  – Conjunction: "Think of a rule that you can apply and involves the conjunction and what you need to prove."

- Disjunction: "Think of a rule that you can apply and involves the disjunction and what you have to prove."

**Notes** The instantiation of the subordinate concept can be derived from the new goal formula.

`Give-away-connect-relevant-subordinate-concept`

**NL Examples**

NL Template: "What connects the < Relevant Concept > and the < Subordinate Concept > is < Rule of Inference >."

- Occurrence state of definitions or substitutions: "What connects the powerset and the subset is the definition of powerset."
- Case distinction: "What connects the disjunctively defined concept *union* and what you have to prove are the cases deriving from the definition of the *union*."
- Induction: "What connects the inductively defined concept *set of all finite subsets of X* and what you have to prove are the inductive steps deriving from its definition."
- Occurrence state of quantifier:
  - Universal: "What connects the *for all* and what you have to prove is the elimination of the *for all*."
  - Existential: "What connects *the exists* and what you have to prove is the elimination of the *there exists*."
- Occurrence state of connective:
  - Equivalence: "What connects the equivalence $\Leftrightarrow$ and what you have to prove is the elimination of the equivalence."
  - Equality: "What connects the equality $=$ and what you have to prove is the elimination of the equality."
  - Implication: "What connects the implication $\Rightarrow$ and what you have to prove is the elimination of the implication."
  - Conjunction: "What connects the conjunction *and* and what you have to prove is the elimination of the conjunction."
  - Disjunction: "What connects the disjunction *or* and what you have to prove is the elimination of the disjunction."

**Notes** The instantiation of the subordinate concept can be derived from the new goal formula.

`Elicit-elaborate-domain-object`

**NL Examples**

NL Template: "Think of a rule that explains how to $<$ Domain Technique $>$ the $<$ Relevant Concept $>$."

- Occurrence state of definitions or substitutions: "Think of rule that explains how to get rid of the powerset."
- Case distinction: "Think of a rule that explains how to apply case distinction to the disjunctively defined concept *union* $\cup$."
- Induction: "Think of a rule that explains how to apply induction to the inductively defined concept *set of all finite subsets of X*."
- Occurrence state of quantifiers:
  - Universal: "Think of a rule that would help you get rid of the *for all*."
  - Existential: "Think of a rule that would help you get rid of the *there exists*."
- Occurrence state of connectives:
  - Equivalence: "Think of a rule that tells you how to get rid of the equivalence $\Leftrightarrow$."
  - Equality: "Think of a rule that tells you how to get rid of the equality $=$."
  - Implication: "Think of a rule that tells you how to get rid of the implication $\Rightarrow$."
  - Conjunction: "Think of a rule that tells you how to get rid of the conjunction *and*."
  - Disjunction: "Think of a rule that tells you how to get rid of the disjunction *or*."

`Give-away-elaborate-domain-object`

**NL Examples**

NL Template: "What helps you $<$ Domain Technique $>$ the $<$ Relevant Concept $>$ is $<$ Rule of Inference $>$."

- Occurrence state of definitions or substitutions: "What helps you get rid of the powerset is the definition of powerset."
- Case distinction: "What helps you apply case distinction to the disjunctively defined *union* $\cup$ are the cases deriving from the definition of the *union*".

- Induction: "What helps you apply induction to the inductively defined concept *set of all finite subsets of X* are the inductive steps deriving from its inductive definition.
- Occurrence state of quantifiers:
  - Universal: "What helps you get rid of the *for all* is the elimination of it."
  - Existential: "What helps you get rid of the *there exists* is the elimination of it."
- Occurrence state of connectives:
  - Equivalence: "What helps you get rid of the equivalence $\Leftrightarrow$ is the elimination of the equivalence."
  - Equality: "What helps you get rid of the equality $=$ is the elimination of the equality."
  - Implication: "What helps you get rid of the implication $\Rightarrow$ is the elimination of the implication."
  - Conjunction: "What helps you get rid of the conjunction *and* is the elimination of the conjunction."
  - Disjunction: "What helps you get rid of the disjunction *or* is the elimination of the disjunction."

`Elicit-inference-rule-application`

**NL Examples**

NL Template: "What do you have to write down instead of the $<$ Relevant Concept $>$ to $<$ Domain Technique $>$ it?"

- Occurrence state of definitions or substitutions: "What do you have to write down instead of the powerset to get rid of it?"
- Case distinction: "What do you have to write down instead of the disjunctively defined *union* $\cup$ to apply case distinction to it?"
- Induction: "What do you have to write down instead of the inductively defined *set of all finite subsets of X* to apply induction to it?"
- Occurrence state of quantifiers:
  - Universal: "What do you have to write down to get rid of the *for all*?"
  - Existential: "What do you have to write down to get rid of the *there exists*?"
- Occurrence state of connectives:

  - Equivalence: “What do you have to write down instead of the equivalence $\Leftrightarrow$ to get rid of it?”
  - Equality: “What do you have to write down instead of the equality $=$ to get rid of it?”
  - Implication: “What do you have to write down instead of the implication $\Rightarrow$ to get rid of it?”
  - Conjunction: “What do you have to write down it instead of the conjunction *and* to get rid of it?”
  - Disjunction: “What do you have to write down instead of the disjunction *or* to get rid of it?”

`Give-away-inference-rule-application`

**NL Examples**

NL Template: “You have to $<$ Domain Technique $>$ the $<$ Relevant Concept $>$ by writing down $<$ Rule of Inference $>$ instead.’

- Occurrence state of definitions or substitutions: “You have to get rid of the powerset by writing down the definition of powerset instead.”
- Case distinction: “You have to apply case distinction for the *union* $\cup$ by writing down the cases deriving from the disjunctive definition of the *union* instead.’
- Induction: “You have to apply induction for *the set of all finite subsets of* $X$ by writing down the inductive steps deriving from its definition instead.”
- Occurrence state of quantifiers:
  - Universal: “You have to get rid of the *for all* by writing down the rule for the elimination of the *for all* instead. ”
  - Existential: “You have to get rid of the *there exists* by writing down the rule for the elimination of the *there exists* instead. ”
- Occurrence state of connectives:
  - Equivalence: “You have to get rid of the equivalence $\Leftrightarrow$ by writing down the rule for the elimination of the equivalence instead.”
  - Equality: “You have to get rid of the equality $=$ by writing down the rule for the elimination of the equality instead.”
  - Implication: “You have to get rid of the implication $\Rightarrow$ by writing down the rule for the elimination of the implication instead.”
  - Conjunction: “You have to get rid of the conjunction *and* by writing down the rule for the elimination of the conjunction instead.”
  - Disjunction: “You have to get rid of the disjunction *or* by writing down the rule for the elimination of the disjunction instead.”

# Appendix E

# Strategy Manager Functions: Main functions, performable-step and meta-reasoning subtasks

In the example functions of the *Strategy Manager* that follow, we use the notation:

- $LH$ – the number of hints produced
- $LC$ – the number of local (current proof step) correct domain contributions
- $LW$ – the number of wrong answers
- $LDCC$ – the number of domain contributions
- $DC$ – the domain contribution
- $DCC$ – the current domain-contribution category
- $PDCC$ – the previous domain-contribution category
- $GMCLA$ – the Global Motivation and Cognitive Load Aggregate
- $PH$ – hint produced already in this proof step
- $STDM$ – the current student task-dialogue-move
- $SBSTR$ – the current substrategy
- $PSTR$ – the previous substrategy

# 1 Main Functions

**Function** `tutoring-control`

Read the analysed student input
Case the student takes something back
    then call `backtracking`
    {*treats diffrent backtrackings*}
Case this is the beginning of the session
    then initiate the task
    else if the proof is completed
        then do recapitulation
Case the proof step is completed
    then reset LMCL and
        if the student completes it
            then accept the answer and
                if GMCLA $\geq 0.3$
                    then call `socratic-generic`
                    {*or any preferred strategy*}
                    else prompt for next step
            else call `socratic-generic`

**Function** `socratic-generic`

Call `encourage`
{*produces appropriate encouragement, if needed*}
Call `signal-evaluation`
{*produces evaluation based on* DCC}
Case $PSTR$ is `near-miss`
    then if $PDCC$ hypotaxis and $PH$ is *give-away-hypotaxis*
    or $PDCC$ is primitive and $PH$ is *give-away-primitive*
    or $PDCC$ is primitive and $PH$ is *give-away-inverse-rule*
    or $DCC$ is step-size and there's been another step-size
        then call `misconception` on near-miss, hypotaxis,
        primitice, inverse-rule, or step-size respectively
Case when inside `misconception`
    then call `misconception` on every
    misconception encountered
    and carry on from the state before `misconception`
Case when inside `spell-out-task`
    then while $LC$ is less than $LDCC$
        then call `spell-out-task`
        else call `conceptual`
Case $PSTR$ is `near-miss`
    then if $PH$ is *discrepancy*
    and $DCC$ is not correct
        then `near-miss`
Case $PSTR$ is `diagnostics` or `request-assistance`
    then call the same substrategy again,
    until what caused it has been dealt with
Case $STDM$ is resign or request-evaluation
    then if there's also a $DC$
        then ignore the $STDM$ and treat the $DC$
        else treat the $STDM$
Case there is a $DC$
    then call `domConCat-output`
    {*produces output based on* DCC}
    else call `studTaskDM-output`
    {*produces output based on* STDM}
If $PSTR$ is not the same as the $SBSTR$
  then signal the closing of $PSTR$
  and the initiation of $SBSRT$

**Function** `conceptual`

Case $LH \leq 2$ and $LW \leq 3$
    call `per-step`
Case $2 < LH \leq 5$ or $LW > 3$
    then if $DC$ is wrong or irrelevant
        then if $LC/DC > 0.5$
        {*at least half answers correct*}
            then call `per-step`
            else give away pending answer,
            produce *give-away-proof-step* and
            *step-meta-reasoning*
    else if $DC$ and *prevDC* are correct
        then call `per-step`
        else reset $LDCC$, $LC$ and call `spell-out-task`
Case $LH > 5$
    then if $DC$ is correct and $LC/DC > 0.5$
        then give away pending answer,
        produce *step-meta-reasoning* and
        *give-away-proof-step*, reset $LDCC$, $LC$
        and call `spell-out-task`
        else call `per-task`
    else give away pending answer,
    produce *step-meta-reasoning* and
    *give-away-proof-step*, *point-to-lesson* and
    reset all counters

# 2 Performable-Step Subtasks

**Subtask** `per-step`

Case student knows the relevant
and the subordinate concept
    if student knows the inference rule
        then if students can substitute
            then call `proof-step`
            else call `substitution`
        else call `inference-rule`
Case student knows the relevant
or the subordinate concept
    then if GMCLA $\leq 0.75$
        then call `domain-object`
        else call `inference-rule`
    else if student knows proof-step meta-reasoning
        then call `domain-object`
        else call `proof-step-meta-reasoning`

**Subtask** `domain-object`

Case student doesn't know the relevant
or subordinate concept
and there's a domain relation used
then if $\leq GMCLA$ 0.5
then call `dom-rel-meta-reasoning`
Case student knows the relevant concept
then if GMCLA $\leq$ 0.5
then produce *elicit-subordinate-concept*
else if GMCLA $\leq$ 0.75
then call `sub-con-meta-reasoning`
Case student knows the subordinate concept
then if GMCLA $\leq$ 0.5
then produce *elicit-relevant-concept*
else GMCLA $\leq$ 0.75
then call `rel-con-meta-reasoning`
Case student knows neither concept
then call `rel-con-meta-reasoning`
Case call `inference-rule`

**Subtask** `proof-step`

if $PH$ is *elicit-proof-step*
then produce *give-away-proof-step* and
*step-meta-reasoning*
else produce *elicit-proof-step*

**Subtask** `inference-rule`

Case if student is missing some basic knowledge
then produce *give-away-basic-knowledge*
Case student knows the subordinate concept
Case student knows the relevant concept
if $PH$ is *elicit-inf-rule*
then if GMCLA $\leq$ 0.5
then produce *give-away-inf-rule*
else call `inf-rule-meta-reasoning`
else produce *elicit-inf-rule*
Case relevant concept not known
then produce *give-away-rel-con*
Case subordinate concept not known
Case student knows the relevant concept
then produce *give-away-sub-con*
Case relevant concept not known
if GMCLA $\leq$ 0.5
then produce *give-away-rel-con*
else call `rel-con-meta-reasoning`

**Subtask** `substitution`

Case $PH$ is *elicit-substitution*
then if the substitution is ill-formed
then if GMCLA > 0.3
{*student is doing very well*}
or $PH$ is *elicit-ill-formed*
then produce *give-away-ill-formed*
else produce *elicit-ill-formed*
else GMCLA > 0.5
then call *subst-meta-reasoning*
else produce *give-away-substitution*
Case produce *elicit-substitution*

# 1 Meta-Reasoning Subtasks

**Subtask** `proof-step-meta-reasoning`

if student does not know the direction or
the directness
then if this is not the first step
or student knows the starting point
{*student knows how to start with the proof*}
then call `all-proof-step-meta-reasoning`
else if this is the first step
and either $PH$ is *elicit-starting-point*
or $GMCLA > 0.75$
then produce *give-away-starting-point*
else produce *elicit-starting-point*

**Function** `all-proof-step-meta-reasoning`

Case student knows premise and conclusion
Case student knows the directness (or direct proof)
then if direction not known
then if $PH$ is *elicit-specific-method*
or GMCLA > 0.75
then produce *give-away-specific-method*
{*gives direction based on infer. rule*}
else produce *elicit-specific-method*
else if direction known
then if prev. hint *elicit-abstract-method*
or GMCLA > 0.75
then produce *give-away-abstr-meth*
else produce *elicit-abstr-method*
Case directness not known
then if direction not known
then if $PH$ is *elicit-specific-method*
or GMCLA > 0.75
then produce *give-away-specific-method*
else produce *elicit-specific-method*
Case premise and conclusion not known
then if $PH$ is *elicit-premise-conclusion*
or GMCLA > 0.75
then produce *give-away-premise-conclusion*
else produce *elicit-premise-conclusion*

**Subtask** `rel-con-meta-reasoning`

Case there's a domain relation used
then call `dom-rel-meta-reasoning`
case the meta-reasoning for relevant is known
or the meta-reasoning for both the relevant and
the subordinate concept is known,
or $GMCLA > 0.75$
then if subordinate concept not known
then produce *give-away-meta-reas-rel-con*
else *give-away-meta-reas-rel-sub-con*
Case no meta-reasoning is known
and $GMCLA \leq 0.75$
then if the subordinate concept is not known
then produce *elicit-meta-reas-rel-sub-Con*
else produce *elicit-meta-reas-rel-Con*

**Subtask** `sub-con-meta-reasoning`

if student knows the relevant concept
  then if $PH$ is *elicit-meta-reas-sub-con*
  or $GMCLA > 0.75$
    then produce *give-away-meta-reas-sub-con*
    else produce *elicit-meta-reas-sub-con*

**Subtask** `inf-rule-meta-reasoning`

Case student uses the inverse of the inference rule
  then *give-away-inverse-rule*
Case student knows the domain technique
  then if $PH$ is *elicit-elaborate-domain-obj*
  or $GMCLA > 0.75$
    then produce *give-away-elaborate-domain-obj*
    else produce *elicit-elaborate-domain-obj*
Case $PH$ is *elicit-connect-rel-sub-con*
or $GMCLA > 0.75$
  then produce *give-away-connect-rel-sub-con*
  else if $PH$ is *elicit-domain-tech*
    then produce *give-away-domain-tech*
    else produce *elicit-domain-tech*
  else produce *elicit-connect-rel-sub-con*

**Subtask** `subst-meta-reasoning`

if $PH$ is *elicit-inf-rule-application*
or $GMCLA > 0.75$
  then produce *give-away-inf-rule-application*
  else produce *elicit-inf-rule-application*

# Appendix F

# Evaluation Study Materials

## F.1 The Original Materials in German

## Beschreibung der Studien

Vorliegende Studie versucht festzustellen, welche Art Feedback am besten geeignet ist, um Studenten beim Erlernen von Beweisen in Mengentheorie zu unterstützen. Ihre Aufgabe besteht darin, das Feedback zu evaluieren. Durch Ihre Bewertung helfen Sie uns, das angemessene Feedback für die jeweilige Unterrichtssituation zu finden. Das Feedback wird Studenten gegeben, während sie Beweise üben, die sie zuvor als Theorie und anhand von Musterlsungen gelernt haben.

## Fragebogen vor der Studie

Bitte füllen Sie die folgenden Informationen aus, bzw. markieren Sie mit “x” die treffende Antwort. Diese Informationen sind nötig für die Datenanalyse. Es gibt keine richtige oder falsche Antwort.

1. Emailadresse: ______________
2. Alter: ______________
3. Geschlect: mänlich / weiblich
4. Ich bin: Student / Lehrer/ Sonstiges (was?) ______________
5. Ich habe veranstaltungen in Mengetheorie besucht: ja / nein
6. Ich habe Mengetheorie unterrichtet: ja / nein

7. Wenn Sie Mengentheorie unterrichtet haben, wo haben Sie sie unterrichtet? Schulle / Nachhilfe/ Sonstiges (was?): ______________

8. Wie würden Sie Ihren Unterrichtsstil beschreiben? ______________

9. Wenn Sie Student sind:

   (a) Ich habe bereits alle Pflichtveranstaltungen in Erziehungswissenschaft besucht: ja / nein

   (b) Wie viele Veranstaltungen in Erziehungswissenschaft haben Sie besucht? ______________

10. Wie betrachtet mann eine Implikation $A \Rightarrow B$?

# Evaluations Fragebogen

## Auswahl Einzelne Rückmeldung

Im Folgenden werden zwei tutorielle Dialoge zwischen einem Tutor **T** und einem Studenten **S** vorgestellt, die den ersten und zweiten Schritt des folgenden Beweises betrachten ($K$ bezeichnet das Komplement):
$A \subseteq K(B) \Rightarrow B \subseteq K(A)$
Nimm an, daß$A \subseteq K(B)$. Wir werden zeigen, dass $B \subseteq K(A)$
Sei belibiges Element $x \in B$.
Wir müssen zeigen, dass auch $x \in K(A)$.
Wenn $x \in B$, dann $x \notin K(B)$ und deshalb, nach der Voraussetzung auch nich in $A$.
Aber wenn $x \notin A$, dann $x \in K(A)$. Es folgt, nach der Definition der $\subseteq$, $B \subseteq K(A)$, weil $x$ belibig ist.

Der Tutor bietet Feedback zu den Lösungsversuchen des Studenten. Für bestimmte Lösungsansätze gibt es zwei Alternativen von Lehrerrückmeldungen (Alternative A und Alternative B). Wir bitten Sie darum, die Alternativen zu vergleichen und durch Unterstreichen zu kennzeichnen, welche der beiden Alternativen Sie bevorzugen. Im Anschluss daran beantworten Sie bitte die Fragen zu Ihren bevorzugten Alternativen, die Sie im Anschluss an jeden tutoriellen Dialog finden. Schließlich bitten wir Sie um eine allgemeine Evaluation des Feedbacks.

#### F.1.0.4 Erster Tutorieller Dialog

**T0:** Hallo!
**S0:** Hallo!
**T1:** Also, sehen wir uns mal einen Beweis an! Erzähl mir alles, was dir einfält in Bezug auf folgenden Ausdruck: Wenn $A \subseteq K(B)$, dann $B \subseteq K(A)$!
**S1:** Ich muss identifizieren, was ich annehmen und was ich beweisen soll.

## Feedback zu S1

**Alternative A**

**T2:** *Das ist nicht vollständig. Zuerst nehmen wir an, dass $A \subseteq K(B)$ gilt, denn das ist die Voraussetzung in dieser Behauptung. Wie ist der nächste Schritt?*

**Alternative B**

**T2:** *Gut! Du bist auf der richtigen Spur. Wir machen es also von Anfang an. Na los, was sollst du denn annehmen und was sollst du beweisen?*

**Ich bevorzuge:** Alternative A Alternative B

**S2:** Ich soll beweisen, dass $B \subseteq K(A)$, und $A \subseteq K(B)$ ist angenommen.

## Feedback zu S2

**Alternative A**

**T3:** *Das ist Rictig! Wie geht's weiter?*

**Alternative B**

**T3:** *Richtig! Nun, wie kannst du den Ausdruck manipulieren, um dein Ziel zu beweisen?*

**Ich bevorzuge:** Alternative A Alternative B

**S3:** Ich muss das Ziel vereinfachen.
**T4:** Richtig! Nun lass uns mal schauen. Versuch etwas im Ausdruck zu finden, das dir helfen kann, das Problem zu vereinfachen.
**S4:** Meinen Sie, die if-dann Beziehung?
**T5:** Genau! Gut, was weißt du über die Wenn-Dann-Beziehung das dir helfen kann, den Ausdruck zu manipulieren?
**S5:** Irgentetwas was ich beweisen muss.

## Feedback zu S5

**Alternative A**

**T6:** *Das ist wieder nicht vollständig. Der Beweis beginnt mit der Annahme, dass $A \subseteq K(B)$ gilt, denn das ist die Voraussetzung. Damit knnen wir die Wenn-Dann-Beziehung auflsen. Wie knnte es damit weitergehen?*

**Alternative B**

**T6:** *Genau! Also, kannst du mir denn sagen, welche Regeln du anwenden musst?*

**Ich bevorzuge:** Alternative A Alternative B

**S6:** Muss ich die Regel anwenden "wenn X ⇒ Y, dann nehmen wir X an und beweisen wir Y"?

## Feedback zu S6

**Alternative A**

**T7:** *Das ist nicht richtig. Wie gesagt, der Beweis beginnt mit der Annahme, dass $A \subseteq K(B)$ gilt, denn das ist die Voraussetzung. Damit können wir die wenn-dann-Beziehung aflösen. Wie könnte es damit weitergehen?*

**Alternative B**

**T7:** *Gut. Nun, versuch den Regeln auf die Ausdrucke anzuwenden.*

**Ich bevorzuge:** Alternative A Alternative B

**S7:** Sei $A \subseteq K(B)$, so wir müssen beweisen, dass $B \subseteq K(A)$.

**T8:** Richtig! Jetzt weiter zum nächsten Schritt!

...

Beantworten Sie jetzt bitte, inwiefern die aufgeführten Begründungen zu Ihrer Wahl passen oder nicht (markieren Sie mit "x" die treffende Antwort). Gehen Sie von der Annahme aus, dass die Studenten ausreichend Erfahrung mit ähnlichen tutoriellen Dialogen haben. Es gibt keine richtige oder falsche Antwort (Ihre Antworten finden Sie auf den Seiten 2 und 3 dieses Dokumentes. Sie können die Miniaturseiten auf der linken Seite benutzen, um schnell zurück zu gehen und sich Ihre Antworte wieder anzusehen. Sie sind hier auf der Seite 4.).

**Die Begründung für meine Wahl zu *S1* ist:**

1. Es ist wahrscheinlicher, dass der Student ein Schema/ Muster lernen wird, um ähnliche Beweissituationen zu bewältigen.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| trifft zu | | | trifft eher zu | | | trifft voll zu |

2. Es ist wahrscheinlicher, dass der Student diese Beweisaufgabe gut lernen wird, auch wenn er kein Schema erworben hat.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| trifft zu | | | trifft eher zu | | | trifft voll zu |

3. Das Feedback ist besser formuliert.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

4. Der Inhalt ist besser.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

5. Es ist wahrscheinlicher, dass der Student von der mitgeteilten Information überfordert wird.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

6. Die mitgeteilte Information ist hilfreicher an dieser Stelle des Dialogs.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

7. Es ist wahrscheinlicher, dass der Student dazu motiviert wird, den Beweis alleine zu schaffen, und besser darauf aufpasst, was der Tutor ihm erklärt.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

8. Es ist wahrscheinlicher, dass der Student nicht versuchen wird, das Feedback des Tutors zu begreifen.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

9. Wenn Ihnen zu den Feedback eine bessere Formulierung einfällt, geben Sie diese bitte hier an:

10. Falls Sie zusätzliche Anmerkungen bezüglich des Feedbacks haben, geben Sie diese bitte hier an:

**Die Begründung für meine Wahl zu *S2* ist:**
...

**Die Begründung für meine Wahl zu *S5* ist:**
...

**Die Begründung für meine Wahl zu *S6* ist:**
...

#### F.1.0.5 Zweiter Tutorieller Dialog

...

**T1:** Lass uns einen Beweis ansehen! Erzähl mir alles, was dir zu folgendem Ausdruck einfällt: Wenn $A \subseteq K(B)$, dann $B \subseteq K(A)$!

**S1:** Nun, beweise B ⊆ K(A), und man muss die wenn-dann Beziehung vereinfachen.

**T2:** Gut. Du bist auf der richtigen Spur. Nun überleg dir, was du über die Wenn-Dann-Beziehung weißt, das dir helfen kann, die richtige Regel für den nächsten Schritt anzuwenden.

**S2:** Meinen Sie, was ich annehmen soll?

**T3:** Okay. Es gibt aber noch eine Sache. Ist das wirklich genau das, was du sagen wolltest?

**S3:** Ja.

**T4:** Nun, das stimmt so eigentlich nicht. Lass uns mal sehen. Was du gesagt hast, ist die Kehrseite von dem, was du brauchst. Weiter gehts!

**S4:** A Teilmenge K(B), daraus folgt B Teilmenge K(A).

## Feedback zu S4

**Alternative A**

**T:** *Das ist noch nicht vollständig, denn du musst noch beweisen, dass B Teilmenge K(A).*

**Alternative B**

**T:** *Das ist ein großer Schritt. Nun lass uns mal sehen. Kannst du zuerst erklären, was du meinst mit "was ich annehmen soll"?*

**S:** *Ich meine, dass A Teilmenge von K(B) ist und dass daraus folgt, dass B Teilmenge von K(A) ist.*

**T:** *Anscheinend gibts hier eine falsche Annahme. Wie gesagt, dieser Schritt ist viel zu groß. Kannst du versuchen, ihn zu erklären?*

**S:** *Ich muss die Wenn-Dann-Beziehung betrachten und dann sagen "sei x und beweise y".*

**T:** *Gut, du machst Forschritte! So, merk dir, dass das, was du annimmst, deine Voraussetzung ist und dass das, was du beweisen musst, deine Folgerung ist. Denk auch daran, dass du alles beweisen musst, was keine Voraussetzung ist oder noch nicht bewiesen ist. Nun versuch die Regel "sei x und beweise y" auf den Ausdruck anzuwenden.*

**Ich bevorzuge:** Alternative A Alternative B

**S5:** Wie kann ich ihn anwenden?

## Feedback zu S5

**Alternative A**

**T:** *Du musst annehmen, dass $A \subseteq K(B)$ und beweisen dass $B \subseteq K(A)$. Wie geht's weiter?*

**Alternative B**

**T:** *Es ist ein wenig schwer, nicht? Wir probieren es mal Schritt für Schritt. Also, sieh zuerst, ob du die Regel auf den Ausdruck anwenden kannst.*

**S:** *Welchen Regel soll ich nehmen?*

**T:** *Die Regel, "wenn $X \Rightarrow Y$, dann nehmen wir $X$ an und beweisen wir $Y$". Kannst du die jetzt anwenden? Na los, versuchs mal!und wend ihn auf die Ausdrucke an. Na los, dann!*

**S:** *Sei $A \subseteq K(B)$, wir müssen beweisen, dass $B \subseteq K(A)$.*

**T:** *Gut, aber deine Antwort ist nicht ganz richtig. Also überleg mal. Du musst die richtigen Teile des Ausdrucks nehmen statt der Variabeln in der Regel. Nun schreib mal den ganzen Schritt aus! S: Sei A Teilmenge K(B), werden wir beweisen, dass B Teilmenge K(A)!*

**S:** *Sei A Teilmenge K(B), wir werden beweisen, dass B Teilmenge K(A)*

**T:** *Richtig! Sehr gut! Also noch einmal, die Beweisführung für diesen Schritt ist folgende: Wir nehmen an, dass A K(B) ist, und müssen beweisen, dass B K(A) ist. Dann, um den Ausdruck zu manipulieren, müssen wir das vereinfachen, was wir beweisen wollen. Also, fangen wir mit der Wenn-Dann-Beziehung an, denn das ist ein Hauptkonzept im Ausdruck und kann uns helfen. Nun müssen wir die Wenn-Dann-Beziehung auflsen, und die Regel, die uns hier weiter bringt, ist "wenn x dann y, nimm x an und beweise y". Dies erreichen wir, indem wir die Variabeln durch die richtigen Teile in unserem Ausdruck ersetzen. Wie kannst du jetzt mit dem nächsten Schritt weitermachen?*

**Ich bevorzuge:** Alternative A Alternative B

**S:** Ich muss einen Regel suchen, mit dem ich dies betrachten kann, stimmt's?

## Feedback zu S6

**Alternative A**

**T:** *Das ist rictig, du musst die Definition der subset benutzen und beweisen, daß wenn $x \in B$, dann $x \in K(A)$.*

**Alternative B**

**T:** *Richtig! Du führst dich gut ein. Also, versuch etwas im Ausdruck zu finden, das dir helfen kann, das Problem zu vereinfachen.*

**S:** *Die Teilmenge.*

**T:** *Richtig! OK, womit ist die Teilmenge verbunden, das dir helfen kann, dein Ziel zu beweisen?*
**S:** *K.*
**T:** *Das stimmt eigentlich nicht. Du kannst das Konzept des Elementes betrachten, und wie das Element mit der Teilmenge verbunden ist, um deinen Ausdruck zu manipulieren. Kannst du weitermachen?*
**Ich bevorzuge:** Alternative A Alternative B

**S7:** Ich hab's! Alle Elemente im B müssen auch Element im K(A) sein.

### Feedback zu S7

**Alternative A**
**T:** *Richtig! Wie kannst du weiter machen?*
**Alternative B**
**T:** *Richtig! Also, lass uns mal wiederholen. Mit welchem Konzept fangen wir an, um den Ausdruck zu vereinfachen?*
**S:** *Teilmenge.*
**T:** *Richtig! Und was war mit der Teilmenge verbunden, das uns helfen kann, den Ausdruck zu manipulieren?*
**S:** *Ja, ja...*
**T:** *OK. Du musst das Element betrachten und dazu benutzen, den Ausdruck zu manipulieren. Nun, wie gehts weiter mit dem nächsten Schritt?*
...

**Ich bevorzuge:** Alternative A Alternative B

Beantworten Sie jetzt bitte, inwiefern die aufgeführten Begründungen zu Ihrer Wahl passen oder nicht (markieren Sie mit "x" die treffende Antwort). Gehen Sie von der Annahme aus, dass die Studenten ausreichend Erfahrung mit ähnlichen tutoriellen Dialogen haben. Es gibt keine richtige oder falsche Antwort (Ihre Antworten finden Sie auf den Seiten 2 und 3 dieses Dokumentes. Sie können die Miniaturseiten auf der linken Seite benutzen, um schnell zurück zu gehen und sich Ihre Antworte wieder anzusehen. Sie sind hier auf der Seite 4.).

**Die Begründung für meine Wahl zu *S4* ist:**
...

**Die Begründung für meine Wahl zu *S5* ist:**
...

**Die Begründung für meine Wahl zu *S6* ist:**
...

**Die Begründung für meine Wahl zu *S7* ist:**
...

## Bewertung des gesamten Feedback

Im Folgenden bitten wir Sie um eine Gesamtevaluation aller Alternativen, die Sie im Einzelnen evaluiert haben. Bitte geben Sie Ihre Gesamtpräferenz an (unterstreichen). Danach beantworten Sie bitte, welche der Begründung auf Ihre Gesamtpräferenz zutrifft (mit "x" markieren). Gehen Sie von der Annahme aus, dass die Studenten ausreichend Erfahrung mit änlichen tutoriellen Dialogen haben. Es gibt keine richtige oder falsche Antwort.

**Insgesamt bevorzuge ich:** Alternativen A Alternativen B

1. Es ist wahrscheinlicher, dass der Student das Gelernte als allgemeine Problemlösungsstrategie auf andere Domänen anwenden wird.

   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

2. Es ist wahrscheinlicher, dass der Student das Gelernte auf andere Probleme in der gleichen Domäne anwenden wird.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

3. Es ist wahrscheinlicher, dass der Student Schemata für Beweise erwerben wird, die nicht explizit unterrichtet werden.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

4. Das Feedback eröffnet dem Studenten mehr Möglichkeiten, die Beweise alleine zu lösen.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

5. Es ist wahrscheinlicher, dass der Student Selbssicherheit gewinnt, um eigenständig zu lernen.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

6. Es ist wahrscheinlicher, dass der Student aufmerksamer mitarbeitet.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

7. Es ist wahrscheinlicher, dass der Student sich aktiv am Lernverfahren beteiligen wird.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

8. Es ist wahrscheinlicher, dass der Student das Gelernte und dessen Wert internalisieren wird.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

9. Es ist wahrscheinlicher, dass der Student den Zusammenhang von verschiedenen Werten, Informationen, und Ideen endecken wird und diese in sein Schema integrieren wird.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

10. Es ist wahrscheinlicher, dass der Student eine durch das Gelernte bewirkte Verhaltensänderung zeigen wird.
    1 2 3 4 5 6 7
    trifft zu trifft eher zu trifft voll zu

11. Es ist wahrscheinlicher, dass der Student den gelernten Stoff im Gedächtnis behält, indem er Fakten, Terminologien, Grundkonzepte, und Antworten, ins Gedächtnis zurückruft.
    1 2 3 4 5 6 7
    trifft zu trifft eher zu trifft voll zu

12. Es ist wahrscheinlicher, dass der Student allgemeine Konzepte und generalisierende Abstraktionen, Prinzipien und Generalisierungen, Theorien und Strukturen in der Domäne erwirbt.
    1 2 3 4 5 6 7
    trifft zu trifft eher zu trifft voll zu

13. Es ist wahrscheinlicher, dass der Student ein Verständnis von Fakten und Ideen aufzeigen wird, indem er in eigenen Worten die Schritte der Durchführung der Aufgabe erklärt.
    1 2 3 4 5 6 7
    trifft zu trifft eher zu trifft voll zu

14. Es ist wahrscheinlicher, dass der Student Probleme durch Verwendung von Domänenkenntnissen, Beweismethoden und Regeln in unterschiedlichen Weisen lsen wird.
    1 2 3 4 5 6 7
    trifft zu trifft eher zu trifft voll zu

15. Es ist wahrscheinlicher, dass der Student Schlussfolgerungen und Beweise finden wird, um Generalisierungen zu machen, und dass er Domänenkenntnisse, Relationen und Organisationsprinzipien analysiert.
    1 2 3 4 5 6 7
    trifft zu trifft eher zu trifft voll zu

16. Es ist wahrscheinlicher, dass der Student Informationen dadurch zusammenstellen wird, dass er Domänenkenntnisse mit neuen Mustern auf verschiedene Weise verknüpft oder alternative Lösungen findet.
    1 2 3 4 5 6 7
    trifft zu trifft eher zu trifft voll zu

17. Es ist wahrscheinlicher, dass der Student durch Urteile und Validierungen von Ideen Meinungen präsentieren und vertreten wird,. z.B. Urteile in Bezug auf interne Beweise.
    1 2 3 4 5 6 7
    trifft zu trifft eher zu trifft voll zu

# Fragebogen nach der Studie

Bitte beantworten Sie zum Abschluss die unten stehenden Aussagen (mit "x" markieren). Es gibt keine richtige oder falsche Antwort.

1. Mir war Schematheorie bereits vor der Studien bekannt.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

2. Mir war Motivationstheorie bereits vor der Studien bekannt.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

3. Mir waren Theorien zu Überforderungen von Lernern bereits vor der Studie bekannt.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

4. Ich stimme Selbstständigem Lernen zu.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

5. Ich kenne mich mit lernerorientiertem Unterricht aus.
   1 2 3 4 5 6 7
   trifft zu trifft eher zu trifft voll zu

6. Ich richte meinen Unterricht am Lerner aus.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| trifft zu | | | trifft eher zu | | | trifft voll zu |

7. Ich bin selbstsicher in meiner Fähigkeit, das Feedback evaluieren zu können.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| trifft zu | | | trifft eher zu | | | trifft voll zu |

8. Ich fühle mich dazu in der Lage, das Feedback in Bezug auf die gestellten Fragen zu evaluieren.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| trifft zu | | | trifft eher zu | | | trifft voll zu |

9. Ich fühle mich dazu in der Lage, das Ziel dieser Evaluationsstudie zu erfüllen.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| trifft zu | | | trifft eher zu | | | trifft voll zu |

10. Ich fühle mich dazu in der Lage, der Aufforderung nachgekommen zu sein, das Feedback in Bezug auf die gestellten Fragen zu evaluieren.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| trifft zu | | | trifft eher zu | | | trifft voll zu |

## F.2 The Experimenter's Materials in English

# Description of the study

In this study, we are trying to find out what is the best feedback to provide to students in order to help them learn proving in set theory. We would like you to rate the feedback as instructed below and help us choose the appropriate feedback each time. This feedback is provided when students are presented with proving tasks to handle as practice on what they have learned in previous learning phases through theory and worked examples.

# Pre-Questionnaire

Please fill in the following information to the best of your knowledge. The information is needed for analysis purposes.

1. Email address: ______________
2. Age: ______________
3. Gender: male /female

4. I am a: university student / teacher/ other (specify)? ______________
5. I have taken courses on Set Theory: yes / no
6. I have teaching experience in Set Theory: yes / no
7. If you have experience, which kind? school / private lessons/ other (specify):
   ______________
8. How would you describe your teaching style?
9. If you are not a teacher:
   (a) I have already completed all compulsory pedagogy modules? yes / no
   (b) How many pedagogy modules have you taken? yes / no
10. How does one prove an implication: $A \Rightarrow B$? ______________

# Evaluation Questionnaires

## Choice of Individual Feedback

You will now read examples of tutorial dialogues between a tutor **T** and a student **S** on the first and second steps of the following proof, where $K$ stands for the complement:

$A \subseteq K(B) \Rightarrow B \subseteq K(A)$
We assume that $A \subseteq K(B)$, as this is the hypothesis, and we prove that $B \subseteq K(A)$ follows.
Then we have to show that all elements of $B$ are also in $K(A)$. Let arbitrary element $x \in B$.
We have to show that $x \in K(A)$ as well.
If $x \in B$, then $x \notin K(B)$ and therefore, according to the assumption also not in $A$.
But if $x \notin A$, then $x \in K(A)$. Therefore, by the definition of $\subseteq$ $B \subseteq K(A)$, because $x$ is arbitrary.

The tutor provides feedback to the student's attempts at the proof. For some student attempts, where it is indicated, we provide two alternatives of tutor feedback (*Alternative A* and *Alternative B*). You are asked to compare and evaluate the alternatives through specific questions, which follow the tutorial dialogue each time. At the end, you are also asked for an overall evaluation of the feedback.

#### F.2.0.6 First Tutorial Dialogue

**T0:** Hello!
**S0:** Hello!
**T1:** OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset K(B), then B is a subset of K(A)!
**S1:** I have to identify what's assumed and what I have to prove.

### Feedback to S1

**Alternative A**

**T2:** *That's not the complete answer. First we assume that $A \subseteq K(B)$ applies, as this is the assumption in this proposition $B \subseteq K(A)$. What's the next step?*

**Alternative B**

**T2:** *Good! You're on the right track. So, we're taking it from the start. Go on then, what is assumed and what do you have to prove?*

**Between the feedback alternatives in *S1* I prefer:**
Alternative A Alternative B

**S2:** I have to prove that $B \subseteq K(A)$, and $A \subseteq K(B)$ is assumed.

### Feedback to S2

**Alternative A**

**T3:** *This is Correct! How can you go on?*

**Alternative B**

**T3:** *Correct! Now, can you manipulate the expression to prove what you want.*

**Between the feedback alternatives in *S2* I prefer:**
Alternative A Alternative B

**S3:** I have to simplify the goal.
**T4:** Correct! OK. Let's see, then. Try to find something in the problem that would help you simplify it.
**S4:** Do you mean the if-then?
**T5:** Exactly! All right then, what do you know about the if-then relation that can help you manipulate the expression.
**S5:** Something about what I need to prove?

## Feedback to S5

**Alternative A**

**T6:** *This is again not the complete answer. The proof starts with the assumption that $A \subseteq K(B)$ applies, as this is the hypothesis. With that, we can break down the if-then relation. How can we go on?*

**Alternative B**

**T6:** *Exactly! So, can then which rule you can use here?*

**Between the feedback alternatives in *S5* I prefer:**
Alternative A Alternative B

**S6:** Do I have to use the rule "if X ⇒ Y, then let X and prove Y"?

## Feedback to S6

**Alternative A**

**T7:** *That is not the complete answer! The proof starts with the assumption that $A \subseteq K(B)$ applies, as this is the hypothesis. With that, we can break down the if-then relation. How can we go on?*

**Alternative B**

**T7:** *Good, that is correct! Now, try to apply the rule to the expression.*

**Between the feedback alternatives in *S6* I prefer:**
Alternative A Alternative B

**S7:** Let $A \subseteq K(B)$ and prove that $B \subseteq K(A)$.
**T8:** Correct! Move on to the next step, now!
...

Now, respond to each of the items that follow in terms of how true it is for you with respect the feedback that you indicate as your preference. Assume sufficient practice through similar tutoring dialogues. There is no right or wrong answer to the questions.

**7-Point Likert Scale:**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| not at all true | | | somewhat true | | | very true |

**The reason for my choice in S1 is:**

1. Schema Promotion
   (a) The student is more likely to learn a schema/pattern for handling similar cases in proving.

   (b) The student is more likely to learn this proof very well, even if he is not able to acquire a schema (reverse item).

2. Feedback Content

   (a) The feedback is better formulated.

   (b) The content is more appropriate.

   (c) The information given away is more likely to overwhelm the student (reverse item).

   (d) The information given away is more appropriate in the context of the dialogue up to this point.

3. Motivation

   (a) The student is more likely to be motivated to complete the proof on his own and pay attention to what the tutor explains to him.

   (b) The student is less likely to try to understand the tutor's feedback. (reversed item)

4. Open Feedback

   (a) If you can think of a more appropriate formulation, please write it here:

   (b) If you have any additional comments on the feedback, please provide them here:

**The reason for my choice in S2 is:**
...

**The reason for my choice in S5 is:**
...

**The reason for my choice in S6 is:**
...

#### F.2.0.7 Second Tutorial Dialogue

...

**T1:** OK, let's look at a proof! Tell me anything you can think of for proving the following: If A is a subset K(B), then B is a subset of K(A)!

**S1:** OK, prove B $\subseteq$ K(A) and simplify the if-then.

**T2:** Good. You are on a good track. Now, consider what you know about the if-then relation that you can help you find the right rule for the next step.

**S2:** Do you mean what I have to assume?

**T3:** Right. There's a minor problem with your answer. Is that really what you wanted to say?

**S3:** Yes.

**T4:** That's not quite right. Let's look at it. What you said is the reverse of what you need. Move on, now!

**S4:** A subset K(B) from that follows B subset K(A).

## Feedback to S4

**Alternative A**

**T:** *That's not complete yet, because you have to prove that B subset K(A).*

**Alternative B**

**T:** *That is big a step. OK. Let's see. Can you first explain what you meant by "what I have to assume" above?*

**S:** *I meant that A is a subset of K(B) and that it follows that B is a subset of K(A).*

**T:** *There seems to be some misconception here. As I say, that is too big a step. Can you explain it?*

**S:** *I have to use if-then and say let "x and prove y".*

**T:** *Good, you're making progress! So, keep in mind that what you assume is your premises and what you have to prove is your conclusion. Also remember that you have to prove or justify everything that is not a premise, or you have not shown before. Now, try apply the rule "let x and prove y" to the expression.*

**Between the feedback alternatives in *S4* I prefer:**
Alternative A Alternative B

**S5:** How do I apply it?

## Feedback to S5

**Alternative A**

**T:** *You have to assume $A \subseteq K(B)$ and prove that $B \subseteq K(A)$. How can you move on?*

**Alternative B**

**T:** *It's a bit difficult, right? Let's try to do this together. Please, see first if you can apply the rule to the expression.*

**S:** *Which rule do I have to use?*

**T:** *Just use the rule if $X \Rightarrow Y$, then let X and prove Y and apply it to the expression. Can you apply it now?*

**S:** *Let A subset K(B), then prove that B subset K(A).*

**T:** *Good! But there's a minor problem with your answer. So, think carefully. You can substitute the appropriate parts of the expression you are dealing with, for the variables in the rule you are applying. Now write the whole step!*

**S:** *Let A subset of K(B), and prove that B subset K(A)*

**T:** *Correct! Very good! The reasoning for this step is as follows: We assume $A \subseteq K(B)$ and what we have to prove is $B \subseteq K(A)$. Next, to manipulate the expression we simplify what we want to prove. So, we start with the if-then relation, because it's central in the problem and can help us simplify what we are trying to prove. Now, we have to get rid of the if-then, and the rule that helps us here is "if $x$ then $y$, assume $x$ and prove $y$". Finally, we do that by substituting the variables by the corresponding values in our expression. Now, how can you start attacking the next step?*

**Between the feedback alternatives in *S5* I prefer:**
Alternative A Alternative B

**S:** I have to look for a rule to handle this, right?

## Feedback to S6

**Alternative A**

**T:** *That's right, you have to use the definition of subset and prove that if $x \in B$, then $x \in K(A)$.*

**Alternative B**

**T:** *Right! That's a good start. Let's see. Try to find something in the problem that would help you simplify the problem.*

**S:** *The subset.*

**T:** *Correct! OK then. Now, what is connected to the subset and can help you prove what you want.*
**S:** *The K.*
**T:** *That's not quite right. You can consider the concept* element*, since you can use it to manipulate the expression. Can you move on?*

**Between the feedback alternatives in *S6* I prefer:**
Alternative A Alternative B

**S7:** I got it! All elements of B should also be elements of K(A).

### Feedback to S7

**Alternative A**
**T:** *Correct! How can you move on?*
**Alternative B**
**T:** *Correct! So, let us look at it again. Which is the concept from which we start simplifying the problem?*
**S:** *The subset.*
**T:** *Correct! And what was connected to the subset that can help you manipulate the expression?*
**S:** *Yeah, yeah...*
**T:** *Ok, you need to consider the element and use it manipulate the expression. What's the next step?*
...

Now, respond to each of the items that follow in terms of how true it is for you with respect the feedback that you indicate as your preference. Assume sufficient practice through similar tutoring dialogues. There is no right or wrong answer to the questions.

**The reason for my choice in S4 is:**
...

**The reason for my choice in S5 is:**
...

**The reason for my choice in S6 is:**
...

**The reason for my choice in S7 is:**
...

## Choice of Overall Feedback

The following statements require an overall evaluation on the feedback alternatives that you just evaluated one by one. Please read all statements once and provide your overall choice of feedback based on them at the end. Then respond to each of them in terms of how true you think it is with respect to your overall preference. For example, if you choose feedback *alternatives A* as your overall preference, then indicate how much you agree that with this feedback as a whole "The student is more likely to apply what he learned in other domains, as a general problem solving technique." and so on for the rest of the statements. Assume sufficient practice through similar tutoring dialogues. There is no right or wrong answer to the questions.

**Overall I prefer the feedback in:** Alternatives A Alternatives B

**7-Point Likert Scale:**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| not at all true | | | somewhat true | | | very true |

**The reason for my choice is:**

1. Distant Transfer
   (a) The student is more likely to apply what he learned in other domains, as a general problem solving technique.
2. Near Transfer
   (a) The student is more likely to apply what he learned in other problems in the same domain.
3. Implicit Learning
   (a) The student is more likely to acquire schemata for theorem proving that are not explicitly taught.
4. Self-Sufficiency
   (a) The feedback gives the student more chances to find the solution alone.
   (b) The student is more likely to become confident in working alone.
5. Evaluation based on Bloom's Taxonomy:
   - Affective

(a) The student is more likely to pay attention. (Receiving)
(b) The student is more likely to participate in the learning process actively. (Responding)
(c) The student is more likely to internalise what is learned and the value of what is learned. (Valuing)
(d) The student is more likely to put together different values, information, and ideas and accomodate them within their own schema, comparing, relating and elaborating on what he learns. (Organising)
(e) The student is more likely to exhibit behaviour that is characterised by what he or she learned. (Characterising)

- Cognitive

(a) The student is more likely to acquire knowledge of facts, terminology, basic concepts, and answers, and to be able to deal with them.
(b) The student is more likely to acquire knowledge of universals and abstractions in the field - principles and generalisations, theories, and structure. (Knowledge)
(c) The student is more likely to demonstrate an understanding of facts and ideas by explaining in their own words the steps of performing the task. (Comprehension)
(d) The student is more likely to solve problems in situations by applying the domain knowledge, proving techniques and rules in different ways. (Application)
(e) The student is more likely to make inferences and find evidence to support generalisations, analyse domain knowledge, relations and organisational principles. (Analysis)
(f) The student is more likely to compile information together in different ways by combining domain knowledge in new patterns or finding alternative solutions, e.g. make a plan and propose a set of operations, derive a set of abstract relations. (Synthesis)
(g) The student is more likely to present and defend opinions by making judgements about information and validity of ideas, e.g., judgements in terms of internal evidence. (Evaluation)

# Post-Questionnaire

Please fill in the following questionnaire. There is no correct answer. The questions aim at helping us analyse the data collected in this study.

**7-Point Likert Scale:**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| not at all true | | | somewhat true | | | very true |

1. Participants background
   (a) I knew about schema theory before.
   (b) I knew about motivation theory before.
   (c) I knew about cognitive load theory before.
   (d) I know what worked examples are.
   (e) I abide by active learning.
   (f) I have had training on student oriented teaching.
   (g) I apply student oriented teaching myself.
2. Perceived Competence to evaluate the feedback
   (a) I feel confident in my ability to evaluate this feedback.
   (b) I am capable of evaluating the feedback based on the asked questions.
   (c) I am able to fulfil the purpose of this evaluation.
   (d) I feel able to meet the challenge of evaluating the feedback based on the asked questions.

-

# Bibliography

Vincent Aleven and Kenneth Koedinger. The need for tutorial dialog to support self-explanation. In Rosé and Freedman [Rosé and Freedman, 2000], pages 65–73.

Vincent Aleven and Kenneth R. Koedinger. Limitations of student control: Do students know when they need help? In G. Gauthier, C. Frasson, and K. VanLehn, editors, *Proceedings of the 5th International Conference on Intelligent Tutoring Systems, ITS 2000*, pages 292–303, Montreal, Canada, 2000.

Vincent Aleven, Kenneth Koedinger, and Karen Cross. Tutoring answer explanations fosters learning with understanding. In S. P. Lajoie and M. Vivet, editors, *Proceeding of World Conference on Artificial Intelligence in Education, AIED-99*, Frontiers in Artificial Intelligence, pages 199–206, Amsterdam, 1999. IOS Press.

Vincent Aleven, Ocav Popescu, and Kenneth R. Koedinger. A tutorial dialogue system with knowledge-based understanding and classification of student explanations. In *Working Notes of 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems.*, Seattle, USA, 2001.

Vincent Aleven, Bruce M. McLaren, Ido Roll, and Kenneth R. Koedinger. Toward meta-cognitive tutoring: A model of help-seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 16:101–130, 2006.

Jan Alexanderson et al. Dialogue acts in verbmobil-2. Technical report verbmobil-report 204, DFKI GmbH Saarbrcken, Universitt Stuttgart, Technische Universitt Berlin,Universitt des Saarlandes, 1997.

James Allen and Mark Core. Draft of DAMSL: Dialogue act markup in several layers. *DRI: Discourse Research Initiative*, University of Pennsylvania, 1997.

James Allen, George Ferguson, and Amanda Stent. An architecture for more realistic conversational systems. In *Proceedings of Intelligent User Interface, IUI01*, pages 14–17, Santa Fe, New Mexico, USA, 2001.

James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. Towards conversational human-computer interaction. *AI magazine*, 2001.

John R. Anderson, Francis S. Belleza, and C. Franklin Boyle. The geometry tutor and skill acquisition. In *Rules of the Mind*. NJ:Erlbaum, 1993.

John R. Anderson, Albert T. Corbett, Kenneth R. Keodinger, and Ray Polletier. Cognitive tutors: Lessons learned. *The Journal of Learning Sciences*, 4:167–207, 1995.

John R. Anderson. *Rules of the Mind*. NJ:Erlbaum, 1993.

Richard Angros, Jr., W. Lewis Johnson, Jeff Rickel, and Schorel Andrew. Learning domain knowledge for teaching procedural skills. In *Proceedings of AAMAS'02*, Bologna, Italy, 2002.

Kevin D. Ashley, Ravi Desai, and John M. Levine. Teaching case-based argumentation concepts using dialectic arguments vs. didactic explanations. In S. A. Cerri, G. Gouardéres, and F. Paraguaçu, editors, *Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, pages 585–595. Berlin: Springer Verlag. A. L. Brown and M. J. Kane, 2002.

Harald Aust and Martin Oerder et al. The philips automatic train timetable information system. *Speech Communication*, 17:249–262, 1995.

Serge Autexier and Armin Fiedler. Textbook proofs meet formal logic — the problem of underspecification and granularity. In Michael Kohlhase, editor, *Mathematical Knowledge Management: 4th International Conference, MKM 2005*, number 3863 in LNAI, pages 96–110. Springer Verlag, 2006.

Serge Autexier, Christoph Benzmller, Armin Fiedler, Helmut Horacek, and Bao Quoc Vo. Assertion-level proof representation with under-specification. *Electronic Notes in Theoretical Computer Science*, 93:5–23, 2004.

Serge Autexier, Christoph Benzmller, Dominik Dietrich, and Jrg Siekmann. Resource adaptive processes in automated reasoning systems. In Matthew Crocker and Jrg Siekmann, editors, *Resource Adaptive Cognitive Processes*, LNAI, pages 28–64. Springer, 2008.

R.S.J.d. Baker. Is gaming the system state-or-trait? educational data mining through the multi-contextual application of a validated behavioral model. In *Complete On-Line Proceedings of the Workshop on Data Mining for User Modeling at the 11th International Conference on User Modeling 2007*, pages 76–80, 2007.

Tifanny Barnes and John Stamper. Toward automatic hint generation for logic proofs tutoring using historical student data. In *The 9th International Conference on Intelligent Tutoring Systems (ITS2008)*, pages 373–382, Berlin Heidelberg, 2008. Springer-Verlang.

Tifanny Barnes, John Stamper, Lorrie Lehmen, and Marvin Croy. A pilot study on logic proof tutoring using hints generated form historical student data. In R.S.J.d. Baker, T. Barnes, and J.E. Beek, editors, *Educational Data Mining 2008: 1st International Conference on Educational Data Mining*, pages 197–201, Montréal, Québec, Canada, 2008.

D. Barrow, A. Mitrovic, S. Ohlsson, and M. Grimley. Assessing the impact of positive feedback in constraint-based tutors. In *The 9th International Conference on Intelligent Tutoring Systems (ITS2008)*, pages 23–27, Montreal, Canad, 2008.

Carole R. Beal and Hyokyeong Lee. Creating a pedagogical model that uses student self reports of motivation and mood to adapt its instruction. In *AIED 05 Workshop on Motivation and Affect in Educational Software*, pages 39–46, Amsterdam, 2005.

C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMEGA: Towards a mathematical assistant. In W. McCune, editor, *Proceedings of the 14th Conference on Automated Deduction*, number 1249 in LNAI, pages 252–255. Springer, 1997.

Christoph Benzmüller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayová, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, and Magdalena Wolska. Tutorial dialogs on mathematical proofs. In *Proceedings of the IJCAI Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, pages 12–22, Acapulco, 2003.

Christoph Benzmüller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayová, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, and Magdalena Wolska. A wizard-of-oz experiment for tutorial dialogues in mathematics. In Vincent Aleven, Ulrich Hoppe, Judy Kay, Riichiro Mizoguchi, Helen Pain, Felisa Verdejo, and Kalina Yacef, editors, *AIED2003 — Supplementary Proceedings of the 11th International Conference on Artificial Intelligence in Education*, volume VIII: Advanced Technologies for Mathematics Education, pages 471–481, Sidney, Australia, 2003. School of Information Technologies, University of Sydney.

N.O. Bernsen, H. Dybkjær, and L. Dybkjær. *Designing Interactive Speech Systems — From First Ideas to User Testing*. Springer, 1998.

D. Berry and D. Broadbent. On the relationship between task performance and the associated verbalizable knowledge. *Quarterly Journal of Experimental Psychology*, 36(A):209–231, 1984.

D. Berry and D. Broadbent. Interactive tasks and the implicit-explicit distinction. *British Journal of Psychology*, 79:251–272, 1988.

B. S. Bloom. *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive Domain.* Longmans, Green, NewYork, 1956.

B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13:4–26, 1984.

Peter Bohlin, Robin Cooper, Elisabeth Engdahl, and Larsson Staffan. Information states and dialogue move engines. In Jan Alexanderson, editor, *Proceedings of the IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue systems*, 1999.

Mark Buckley and Christoph Benzmüller. System Description: A Dialogue Manager supporting Natural Language Tutorial Dialogue on Proofs. In David Aspinall and Christoph Lüth, editors, *Proceedings of the ETAPS Satellite Workshop on User Interfaces for Theorem Provers (UITP)*, pages 40–67, Edinburgh, Scotland, 2005.

Mark Buckley and Christoph Benzmüller. A Dialogue Manager supporting Natural Language Tutorial Dialogue on Proofs. *Electronic Notes in Theoretical Computer Science*, 2006. To appear.

Richard R. Burton. Diagnosing bugs in a simple procedural skill. In D. Sleeman and S. J. Brown, editors, *Intelligent Tutoring Systems*, Computers and Poeple, pages 157–183. Academic Press, 1982.

Ch. Callaway, D. Dietrich, M. O. Dzikovska, , E. Farrow, M. Homik, M Marques-Pita, C. Matheson, E. Melis, J. D. Moore, and C. Ullrich. Refined nlg and nlu. Technical report, The consortium collaborators, 2006.

G. Carenini and J. Moore. Generating and evaluating evaluative arguments. *Artificial Intelligence Journal*, 170:925–952, 2006.

Paul Chandler and John Sweller. Cognitive load theory and the format of instruction. *Cognition and Instruction*, 8(4):293–332, 1991.

M. T. H. Chi, R. Glaser, and E. Rees. Expertise in problem solving. *Advances in the Psychology of Human Intelligence*, pages 7–75, 1982.

Michelene T. H. Chi, Matthew W. Lewis, Peter Riemann, and Robert Glaser. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13:145–182, 1989.

Michelene T. H. Chi, Nicholas de Leeuw, Mei-Hung Chiu, and Christian Lavancher. Eliciting self-explanation improves understanding. *Cognitive Science*, 18:439–477, 1994.

Allan Collins and Albert L. Stevens. Goals and strategies of inquiry teachers. *Advances in Instructional Psychology*, 2:65–119, 1982.

A. Collins and A. L. Stevens. A cognitive theory of inquiry teaching. In P. Goodyear, editor, *Teaching Knowledge and Intelligent Tutoring*, pages 203–230. Ablex Publishing Corporation, Norwood, New Jersey, 1991.

Allan Collins. Cognitive apprenticeship and instructional technology. In L. Idol and B. F. Jones, editors, *Educational Values and Cognitive Instruction*, pages 121–138. Erlbaum, Hillsdale, NJ, 1991.

Cristina Conati and Kurt VanLehn. Teaching meta-cognitive skills: implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. In *Proceedings of AIED '99, 9th World Conference of Artificial Intelligence and Education*, pages 297–304, Amsterdam: IOS press, 1999.

Graham Cooper and John Sweller. Effects of schema acquisition and rule automation on mathematical problem transfer. *Journal of Educational Psychology*, 79(4):347–362, 1987.

Robin Cooper, Staffan Larsson, Colin Matheson, Massimo Poesio, and David Traum. Coding instructional dialogue for information states. Technical report, University of Gothenburg, 1999.

A. Corbett, K.R. Koedinger, and J.R. Anderson. Intelligent tutoring systems. In M. Helaner and T.K. Landauer, editors, *Handbook of Human-Computer Interaction, Second Edition*, pages 849–874. Amsterdam: Elsevier Science, 1997.

Mark G. Core and James F. Allen. Coding dialogues with DAMSL annotation scheme. In *AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35, Boston, MA, 1993.

Mark G. Core, Johanna D. Moore, and Claus Zinn. Supporting constructive learning with a feedback planner. In *Proceedings of the AAAI Fall Symposium: Building Dialogue Systems for Tutorial Applications*, Falmouth, MA, 2000. AAAI Press.

Mark G. Core, Johanna D. Moore, Claus Zinn, and Peter Wiemer-Hastings. Modeling human teaching tactics in a computer tutor. In *Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies National ACM conference, NAACL-2001, Workshop on Adaptation in Dialogue Systems*, Pittsburgh, PA, 2001.

Mark G. Core, Johanna D. Moore, and Claus Zinn. Draft: Tutorial annotation scheme. September 11, 2002.

A. de Vicente and H. Pain. Motivation diagnosis in intelligent tutoring systems. In *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems*, volume 1452 of Lecture Notes in Computer Science, pages 86–95, Berlin, 1998. Springer.

Victor R. Delclos and Christine Harrington. Effects of strategy monitoring and proactive instruction on children's problem solving performance. *Journal of Educational Psychology*, 83(1):35–42, 1991.

Vania Dimitrova, John Self, and Paul Brna. The interactive maintenance of open learner models. In S.P. Lajoie and M. Vivet, editors, *In Artificial intelligence in education*, pages 405–412. IOS Press, Amsterdam, 1999.

R.E. DiPaolo, A.C. Graesser, H.A. White, and D.J. Hacker. Hints in human and computer tutoring. In M. Rabinowitz, editor, *The design of instruction and evaluation*, pages 155–182. Mahwah, NJ: Erlbaum, 2004.

M. Driscoll. *Psychology of Learning for Instruction*. Allyn and Bacon, Boston, 1994.

Benedict du Boulay and Rosemary Luckin. Modelling human teaching tactics and strategies for tutoring systems. *International Journal of Artificial Intelligence in Education*, 12:235–256, 2001.

M. O. Dzikovska, C. B. Callaway, E. Farrow, M. Marques-Pita, C. Matheson, and J. D. Moore. Adaptive tutorial dialogue systems using deep nlp techniques. In *Proceedings of NAACL-2007 Demo Session*, Rochester, NY, USA, 2007.

Myroslava Dzikovska, Gwendolyn Campbell, Charles Callaway, Natalie Steinhauser, Elaine Farrow, Johanna Moore, Leslie Butler, and Colin Matheson. Diagnosing natural language answers to support adaptive tutoring. In *Proceedings of the 21st FLAIRS Conference*, Miami, Florida, 2008.

Martha W. Evens, Stefan Brandle, Ru-Charn Chang, Reva Freedman, Michael Glass, Leeand Leem Seop Shim Yoon Hee, Chong Woo Woo, Yuemei Zhang, Zhou Yujian, Joel A. Michael, and Allen A. Rovick. Circsim-Tutor: An intelligent tutoring system using natural language dialogue. In *Proceedings Twelfth Midwest AI and Cognitive Science Conference, MAICS 2001*, pages 16–23, Oxford, 2001.

Michael W. Eysenck and Mark T. Keane. *Cognitive Psychology: A student's handbook*. Psychology Press, USA and Canada, 4th edition, 2000.

G. Ferguson and J. Allen. Mixed-initiative dialogue systems for collaborative problem-solving. *AI Magazine: Special Issue on Mixed-Initiative Assistants*, 28(2):23–32, 2007.

Armin Fiedler and Malte Gabsdil. Supporting pogressive refinement of Wizard-of-Oz experiments. In Carolyn Penstein Rosé and Vincent Aleven, editors, *Proceedings of the ITS 2002 — Workshop on Empirical Methods for Tutorial Dialogue Systems*, pages 62–69, San Sebastián, Spain, 2002.

Armin Fiedler and Dimitra Tsovaltzi. Automating hinting in an intelligent tutorial system. In *Proceedings of the IJCAI Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems*, pages 23–35, Acapulco, 2003.

Armin Fiedler and Dimitra Tsovaltzi. Automating hinting in mathematical tutorial dialogue. In *Proceedings of the EACL-03 Workshop on Dialogue Systems: Interaction, Adaptation and Styles of Management*, pages 45–52, Budapest, 2003.

Armin Fiedler and Dimitra Tsovaltzi. Domain-knowledge manipulation for dialogue-adaptive hinting. In *12th International Conference on Artificial Intelligence in Education (AIED 2005)*, Amsterdam, 2005.

Armin Fiedler, Andreas Franke, Helmut Horacek, Markus Moschner, Martin Pollet, and Volker Sorge. Ontological issues in the representation and presentation of mathematical concepts. In Jérôme Euzenat, Asuncion Gómez Pérez, Nicola Guarino, and Heiner Stuckenschmidt, editors, *Proceedings of the ECAI 2002 Workshop on Ontologies and Semantic Interoperability*, pages 62–66, Lyon, France, 2002.

A. Fiedler. Dialog-driven adaptation of explanations of proofs. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1295–1300, Seattle, WA, 2001. Morgan Kaufmann.

A. Fiedler. *User-adaptive proof explanation*. PhD thesis, Department of Computer Science, Universität des Saarlandes, Saarbrücken, Germany, 2001.

Reva Freedman, Zhou Yujian, Michael Glass, Jung Hee Kim, and Martha W. Evens. Using rule induction to assist in rule construction for a natural-language based intelligent tutoring system. In *Proceedings Twentieth Annual Conference of the Cognitive Science Society*, pages 362–367, Madison, 1998.

Reva Freedman. Plan-based dialogue management in a physics tutor. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP '00)*, Seatle, 2000.

Claudia Gama. Metacognition in interactive learning environments: The reflection assistant model. In ITS2004 [ITS2004, 2004], pages 668–677.

Abigail S. Gertner, Christina Conati, and Kurt VanLehn. Procedural help in Andes: Generating hints using a bayesian network student model. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 106–111, Madison, Wisconsin, 1998. AAAI Press/MIT Press.

Jonathan Ginzburg. An update semantics for dialogue. In *Proceedings of the International Workshop on Computational Semantics*, Tilburg, 1994.

Jonathan Ginzburg. Interrogatives: Questions, facts and dialogue. In *The Handbook of Contemporary Semantic Theory*. Blackwell, Oxford, 1996.

Susan R. Goldman. On the derivation of instructional applications from cognitive theories: Commentary on chandler and sweller. *Cognition and Instruction*, 8(4):333–342, 1991.

A.C. Graesser, K. Wiemer-Hastings, P. Wiemer-Hastings, R. Kreuz, and Tutoring Research Group U.o.M. Autotutor: A simulation of a human tutor. *Cognitive Systems Research*, 1:35–51, 1999.

Arthur C. Graesser, Natalie K. Person, Derek Harter, and the Tutoring Research Group. Teaching tactics and dialogue in autotutor. *International Journal of Artificial Intelligent in Education*, 2001.

A.C. Graesser, K. Moreno, J. Marineau, A. Adcock, A. Olney, and N. Persons. Autotutor improves deep learning of computer literacy: Is it the dialog or the talking head? In *Proceedings of the Conference on Artificial Inteligence in Education*, pages 47–54, Sydney, 2003.

A.C. Graesser, S. Lu, G.T. Jackson, H. Mitchell, M. Ventura, A. Olney, and M.M. Louwerse. Autotutor: A tutor with dialogue in natural language. *Behavioral Research Methods, Instruments, and Computers*, 36:180–193, 2004.

A.C. Graesser, D.S. McNamara, and K. VanLehn. Scaffolding deep comprehension strategies through point&query, autotutor, and istart. *Educational Psychologist*, 40:225–234, 2005.

J. G. Greeno. Natures of problem solving abilities. In William K. Estes, editor, *Handbook of learning and cognitive processes*, volume 5. Hillsdale, N.J., 1978.

R. De Groot, R. Drachman, R. Hever, B. Schwarz, A. Harre U. Hoppe, M. De Laat, R. Wegerif, B. M. McLaren, and B. Baurens. Computer supported moderation of e-discussions: the argunaut approach. In *Proceedings of the Conference on Computer Supported Collaborative Learning (CSCL-07)*, pages 165–167, 2007.

B.J. Grosz and C.L. Sidner. Attention, intention and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

Neil T. Heffernan and Kenneth R. Koedinger. A development model for algebra symbolization: The results of a difficulty factors assessment. In *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*, pages 484–489, Hillsdale, NJ, 1998. Lawrence Erlbaum Associates.

Helmut Horacek and Magdalena Wolska. Handling errors in mathematics formulas. In Tak-Wai Chan Mitsuru Ikeda, Kevin D. Ashley, editor, *Proceedings of the 8th International Conference, ITS 2006*, pages 337 –348, Jhongli, Twaiwan, 2006. Springer.

Helmut Horacek and Magdalena Wolska. Interpreting semi-foormal utterances in dialoggs about mathematical proofs. *Data and Knowledge Engineering*, 58:90–106, 2006.

Helmut Horacek. Up-to-the-point hints in tutoring mathematical theorem proving. In *Cybernetic and Systems Vol 2, Proceedings of the 18th European Meeting on Cybernetic and System Research, Austrian Society for Cybernetic Studies*, 2006.

Joris Hulstijn. Roles in dialoguemarked. In Ivana Kruijff-Korbayová and Claudia Kosny, editors, *Proceedings of DiaBruck'03, the 7th Workshop on the Semantics and Prgmatics of Language*, Saarbrücken, Germany, 2003.

Gregory Hume, Joel Michael, Allen Rovick, and Martha Evens. Student responses and follow up tutorial tactics in an ITS. In *Proceedings of the 9th Florida Artificial Intelligence Research Symposium*, pages 168–172, Key West, FL, 1996.

Gregory D. Hume, Joel A. Michael, Allen A. Rovick, and Martha W. Evens. Hinting as a tactic in one-on-one tutoring. *Journal of the Learning Sciences*, 5(1):23–47, 1996.

Gregory D. Hume. *Using Student Modelling to Determine When and How to Hint in an Intelligent Tutoring System.* PhD thesis, Illinois Institute of Technology, Chicago, IL, 1995.

*Intelligent Tutoring Systems – 7th International Conference, ITS 2004*, LNCS. Springer, 2004.

Pamela W. Jordan and Kurt VanLehn. Discourse processing for explanatory essays in tutorial applications. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue*, Philadelphia, USA, 2002.

P. Jordan, P. Albacete, and K. VanLehn. Taking control of redundacy in scripted tutorial dialogue. In *Proceedings of International Conference on Artificial Intelligence in Educations (AIED05)*, pages 314–321, 2005.

Pamela Jordan, Maxim Makatchev, Umarani Pappuswamy, Kurt VanLehn, and Patricia Albacet. A natural language tutorial dialogue system for physics. In *Proceedings of the 19th International FLAIRS conference*, 2006.

Elena Karagjosova. Marked informationally redundant utterances in tutorial dialogue. In Ivana Kruijff-Korbayová and Claudia Kosny, editors, *Proceedings of DiaBruck'03, the 7th Workshop on the Semantics and Prgmatics of Language*, Saarbrücken, Germany, 2003.

S. Katz, A. Lesgold, E. Hughes, D. Reters, , G. Eggan, and M. Gordin et al. Sherloch 2: An intelligent tutoring system built on the lrdc frameword. In C.P. Bloom and R.B. Loftin, editors, *Facilitating the development and use of interactive learning environments*, pages 227–258. Hilisdale NJ: Erlbaum, 1998.

J. M. Keller. Strategies for simulating the motivation to learn. *Performance and Instruction*, 26(8):1–7, 1987.

Kenneth R. Koedinger and John R. Anderson. Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science*, 14:511–550, 1990.

Kenneth R. Koedinger and John R. Anderson. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43, 1997.

Michael Kohlhase. *OmDoc – An Open Markup Format for Mathematical Documents [Version 1.2]*, volume 4180 of *LNAI*. Springer, 2006.

Joern Kreutel and Colin Matheson. Modelling questions and assertions in dialogue using obligations. In *Proceedings 3rd International Workshop on the Semantics and Pragmatics of Dialogue, Amstelogue*, University of Amsterdam, 1999.

Joern Kreutel and Colin Matheson. Context-dependent interpretations and implicit dialogue acts. In *BI-DIALOG*, Bielfeld, 2001.

H. Chad Lane and K. VanLehn. Teaching the tacit knowledge of programming to novices with natural language tutoring. *Computer Science Education, Special Issue on Doctoral Research in Computer Science Education*, 15(3):183–201, 2005.

Staffan Larsson and Robin Cooper. An information state approach to natural interactive dialogue. In *Proceedings LREC2000 Workshop on Natural Interactive Dialogue*, 2000.

Staffan Larsson, Peter Bohlin, Johan Bos, and David Traum. *Trindikit 1.0 manual*. University of Gothenburgh, 1999.

Staffan Larsson, Gabriel Amores, Elena Karagjosova, David Milward, and Dimitra Tsovaltzi. Flexible dialogue. Technical report siridus project deliverable d1.4, University of Gothenburgh, 2002. Distribution: PUBLIC.

Staffan Larsson. *Issue-based Dialogue Management*. PhD thesis, Department of Linguistics, Goeteborg University, Sweden, 2002.

Chung Hee Lee, Jai Hyun Seu, and Martha W. Evens. Building an ontology for CIRCSIM-Tutor. In *Proceedings of the 13th Midwest AI and Cognitive Science Conference, MAICS-2002*, pages 161–168, Chicago, 2002.

A. Lesgold, S. Lajoie, M. Bunzo, and G. Eggan. Sherlock: A coached practice environment for an electronics troubleshooting job. In J. Larkin and R. Chabay, editors, *Computer-assised instruction and intelligent tutoring systems. Shared goals and complementary approaches*. Hilisdale, NJ: Lawrence Erlbaum Associates, 1992.

Pawel Lewicki, Thomas Hill, and Maria Czyzewska. Nonconscious acquisition of information. *Journal of American Psychologist*, 47:796–801, 1992.

Eng Leong Lim and Dennis W. Moore. Problem solving in geometry: Comparing the effects of non-goal specific instruction and conventional worked examples. *Journal of Educational Psychology*, 22(5):591–612, 2002.

E. L. Lim, R. S. Dixon, and D. W. Moore. Worked examples vs. non goal-specific problems: A test of schema development in geometry. *Educational Psychology*, 16:421–431, 1996.

A. H. Maslow. A theory of human motivaton. *Psychological Review*, 50:370–396, 1943.

Colin Matheson, Massimo Poesio, and David Traum. Modelling grounding and discourse obligations using update rules. In *Proceedings NAACL 2000*, Seattle, 2000.

R. Mathews, R. Buss, W. Stanley, F. Blanchard-Fields, J. Cho, and N. Druhan. Role of implicit and explicit processes in learning from examples: a synergistic effect. *Journal of Experimental Psychology*, 15:1083–1100, 1989.

Noboru Matsuda and Kurt VanLehn. Modelling hinting strategies for geometry theorem proving. In P. Brusilovsky, A. Corbett, and F. de Rosis, editors, *Proceedings of the 9th International Conference on User Modeling*, pages 373–377, Pittsburgh, PA, 2003. Berlin, Heidelberg: Springer.

Noboru Matsuda and Kurt VanLehn. Advanced geometry tutor: An intelligent tutor that teaches proof-writing with construction. In C. K. Looi et al., editor, *Artificial Intelligence in Education*, pages 443–450. IOS Press, 2005.

Manolis Mavrikis, Antony Maciocia, and John Lee. Targeting the affective state of students studying mathematics in a web-based ile. In *Proceedings of AIED2003 Conference of Artificial Intelligence in Education*, pages 77–82, Sydney, 2003.

Richard E. Mayer, W. Lewis Johnson, Erin Shaw, and Sahiba Sandhu. Constructing computer-based tutors that are socially sensitive: Politeness in educational software. *Int. J. Human-Computer Studies*, 64(1):36–42, 2006.

B.M. McLaren, S. Lim, F. Gagnon, D. Yaron, and K.R. Koedinger. Studying the effects of personalized language and worked examples in the context of a web-based intelligent tutor. In *Proceedings of the the 8th International Conference on Intelligent Tutoring Systems (ITS-2006)*, pages 318–328, Jhongli, Taiwan, 2006. IOS Press.

B.M. McLaren, S. Lim, D. Yaron, and K.R. Koedinger. Can a polite intelligent tutoring system lead to improved learning outside of the lab? In *Proceedings of the of the 13th International Conference on Artificial Intelligence in Education (AIED 2007)*, pages 443–440, Amsterdam, The Netherlands, 2007. IOS Press.

Bruce M. McLaren, Sung-Joo Lim, and Kenneth R. Koedinger. When and how often should worked examples be given to students? new results and a summary of the current state of research. In V. M. Sloutsky B. C. Love, K. McRae, editor, *Proceedings of the 30th Annual Conference of the Cognitive Science Sociaty*, pages 2176–2181, Austin, TX, 2008. Cognitive Science Society.

M. F. McTear. Modelling spoken dialogues with state transition diagrams: Experiences with the cslu toolkit. In *Proceedings of the 5th International Conference on Spoken Language Processing*, Sydney, Australia, 1998.

Chris Mellish. Modelling politeness in natural language generation. In *Proceedings of the International Natural Language Generation Conference*, 2004.

Goerge A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

Marvin Minsky. A framework for representing knowledge. *The Psychology of Computer Vision*, 1975.

Antonija Mitrovic and Stellan Ohlsson. Evaluation of a contraint-based tutor for a database language. *International Journal of Artificial Intelligence in Education*, 10:238–256, 1999.

Antonija Mitrovic. An intelligent sql tutor on the web. *International Journal of Artificial Intelligence in Education*, 13(2-4):197–243, 2003.

Johanna Moore. What makes human explanations effective? In *Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society*, Hillsdale, NJ, 1993. Lawrence Erlbaum Associates.

Johanna et al Moore. The BE&E corpus, 2000. `www.hcrc.ed.ac.uk/~jmoore/tutoring/BEE_corpus.html`.

C. Murray, K. VanLehn, and J. Mostow. Looking ahead to select tutorial actions: A decision-theoretic approach. *International Journal of Artificial Intelligence in Education*, 14(3-4):235–278, 2004.

T. Murray. Eon: Authoring tools for content, instructional strategy, student model, and interface design. In T. Murray S. Blessing and S. Ainsworth, editors, *Authoring Tools for Advanced Technology Learning Environments.* Kluwer Academic/Springer, Netherlands, 2003.

Susanne Narciss. Feedback strategies for interactive learning tasks. In J. M. Spector, M. D. Merrill, J. J. G. van Merrinboer, and M. P. Driscoll, editors, *Handbook of Research on Educational Communications and Technology*, pages 125–143. Mahwah, NJ: Lawrence Erlbaum, 2003. 3rd ed.

M.J. Nathan and K.R. Koedinger. Teachers' and researchers' beliefs of early algebra development. *Journal for Research in Mathematics Education*, 31(2):168–190, 2000.

E. Owen and J. Sweller. What do students learn while solving mathematical problems? *Journal of Educational Psychology*, 77:272–284, 1985.

Fred Paas, Alexander Renkl, and John Sweller. Cognitive load theory: Instructional implications of the interaction between information structures and cognitive architecture. *Instructional Science*, 32(1-8), 2004.

Natalie K. Person and Arthur Graesser. Fourteen facts about human tutoring: Food for thought for ITS developers. In *aied2003, Supplementary Procceedings*, pages 355–344, Sydney, 2003.

Natalie K. Person, Arthur C. Graesser, Derek Harter, Eric Mathews, and the Tutoring Research Group. Dialog move generation and conversation management in AutoTutor. In Rosé and Freedman [Rosé and Freedman, 2000], pages 45–51.

Massimo Poesio and David Traum. Towards an axiomatisation of dialogue acts. In J. Hulstij and A. Nijholt, editors, *Proceedings Twentieth Workshop on the Formal Semantics and Pragmatics of Dialogues*, pages 207–222, Enschede, 1998.

K. Porayska-Pomsta and C. Mellish. Modelling politeness in natural language generation. In Srpinger, editor, *Proceedings of the 3rd International Conference on Natural Language Generation*, pages 141–150, 2004.

K. Porayska-Pomsta and C. Mellish. Determining tutorial remediation strategies from a corpus of human-human tutoring dialogues. In *Proceedings of the 11th European Workshop on Natural Language Generation (ENLG)*, pages 123–130, Schloss, Dagstuhl, 2007.

K. Porayska-Pomsta and C. Mellish H. Pain. Aspects of speech act categorisation: Towards generating teachers' language. *International Journal of Artificial Intelligence in Education*, 11, 2000.

K. Porayska-Pomsta and H. Pain. Providing cognitive and affective scaffolding through teaching strategies. In Srpinger, editor, *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS2004)*, pages 77–86, 2004.

E. Price, Cotton, and R. Klatzky M. Priscoll. An inquiry into the spontaneous transfer of problem solving skill. In *Contemporary Educational Psychology*, volume 22 (4), pages 472–494, 1997.

S. Rajan, S.D. Craig, B. Gholson, N.K. Person, and A.C. Graesser. Autotutor: Incorporating back-channel feedback and other human-like conversational behaviors into and intelligent tutoring system. *International Journal of Speech Technology*, 4(2):117–126, 2001.

Genaro Rebolledo-Méndez. Modelling the motivational state of the learner in a vytotskyan inspired its. In *Proceedings of AIED2003 Conference of Artificial Intelligence in Education*, pages 95–100, Sydney, 2003.

Chris Reed and Floriana Grasso, editors. *Special Issue on Computational Models of Natural Argument of the International Journal of Intelligent Systems (IJIS).* John Wiley & Sons, 2004.

K. Reiss, M. Moormann, Ch. Gro, and C. Ullrich. Formalized pedagogical strategies. Technical report, Universität of Augsburgh and Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 2005.

Charles Rich and Candace L. Sidner. Collagen: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3/4):315–350, 1998.

Jeff Rickel, Rajaram Ganeshan, Charles Rich, Candance L. Sidner, and Neal Lesh. Task-oriented tutorial dialogue: Issues and agents. In Rosé and Freedman [Rosé and Freedman, 2000], pages 52–57.

Michael A. Ringenberg and Kurt VanLehn. Scaffolding problem solving with annotated workrd-out examples to promote deep learning. In Tak-Wai Chan Mitsuru Ikeda, Kevin D. Ashley, editor, *Proceedings of the 8th International Conference, ITS 2006*, pages 625–634, Jhongli, Twaiwan, 2006. Springer.

Ian Robertson. Imitating problem solving: Why transfer of learning often fails to occur. *Instructional Science*, 28:263–289, 2000.

Ido Roll, Vincent Aleven, Bruce M. McLaren, Eunjeong Ryu, Ryan Baker, and Kenneth R. Koedinger. The help-tutor: Does metacognitive feedback improve students' help-seeking actions, skills and learning? In Srpinger Verlang, editor, *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS2006)*, pages 360–369, Berlin, 2006.

I. Roll, V. Aleven, B. M. McLaren, , and K. R. Koedinger. Can help seeking be tutored? searching for the secret sauce of metacongitive tutoring. In *International Conference on Artificial Intelligence in Education 2007*, Los-Angeles, CA, 2007.

Carolyn Penstein Rosé and Reva Freedman, editors. *Building Dialog Systems for Tutorial Applications—Papers from the AAAI Fall Symposium*, North Falmouth, MA, 2000. AAAI press.

C. P. Rosé, P. Jordan, M. Ringenberg, S. Siler, Kurt VanLehn, and Anders Weinstein. Interactive conceptual tutoring in atlas-andes. In *Proceedings of AI in Educations 2001 Conference (AIED01)*, 2001.

Carolyn P. Rosé, Johanna D. Moore, Kurt VanLehn, and David Allbritton. A comparative evaluation of socratic versus didactic tutoring. In Johanna Moore and Keith Stenning, editors, *Proceedings 23rd Annual Conference* *of the Cognitive Science Society*, pages 869–874, University of Edinburgh, Scotland, UK, 2001.

Carolyn Perstein Rosé, Johanna D. Moore, Kurt VanLehn, and David Allbritton. A comparative evaluation of socratic versus didactic tutoring. In *Proceedings 23rd Annual Conference of the Cognitive Science Society*, Edinburgh, Scotland, UK, August 2001.

Michael R. Ross and Robert B. Fulton. Active learning strategies in the analytical chemistry classroom. *Journal of Chemical Education*, pages 141–143, 1994.

D.E. Rumelhart and A. Ortony. The representation of knowledge in memory. In R.J. Spiro R.C. Anderson and W.E. Montauge, editors, *Schooling and the acquisition of knowledge*. Lawrence Erlbaum Associates Inc., Hillsdale, NJ, 1977.

D.E. Rumelhart, P. Smolensky, J.L. McClellant, and G.E. Hinton. Schemata and sequential thought processes in pdp models. In J.L. McClellan, D.E. Rumelhart, and PDP Research Group, editors, *Parallel distributed processing: Psychological and biological models*, volume 2. MIT Press, Cambridge, MA, 1986.

D.E. Rumelhart. Schemata: The basic building blocks of cognition. In B. Bruce R. Spiro and W. Brewer, editors, *Theoretical issues in reading comprehension*. Lawrence Erlbaum Associated Inc., Hillsdale, NJ, 1980.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Inc., N.J., 2003. 2nd edition.

Roger C. Schank and Robert P. Abelson. *Scripts, plans, goals and understanding: an inquiry into human knowledge structures*. Erlbaum, Hillsdale, N. J., 1977.

Marvin Schiller, Christoph Benzmüller, and Ann Van de Veire. Judging granularity for automated mathematics teaching. In *LPAR 2006 Short Papers Proceedings*, Phnom Pehn, Cambodia, 2006.

A. H. Schoenfeld and D. J. Herrmann. Problem perception and knowledge structure in expert and novice mathematical problem solvers. *Journal of Experimental Psychology*, 8:484–494, 1982.

Wolfgang Schreiner. Mathematics 2 (Formal Proving). Technical report, University of Applied Sciences at Hagenberg, 2004.

J. R. Searle. A taxonomy of illocutionary acts. *Language, Mind and Knowledge, Minnesota Studies in the Philosophy of Science*, pages 344–369, 1975.

Nicole Shechtman and Leonard M. Horowitz. Media inequality in conversation: how people behave differently when interacting with computers and people. In *Proceedings of the CHI 2003 conference on human factors in computing systems*, pages 5–10, Florida, USA, 2003.

Jörg Siekmann, Christoph Benzmüller, Vladimir Brezhnev, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhase, Andreas Meier, Erica Melis, Markus Moschner, Immanuel Normann, Martin Pollet, Volker Sorge, Carsten Ullrich, Claus-Peter Wirth, and Jürgen Zimmer. Proof development with ΩMEGA. In Andrei Voronkov, editor, *Automated Deduction — CADE-18*, number 2392 in LNAI, pages 144–149. Springer Verlag, 2002.

John Stamper. Automating the generation of student models for intelligent tutoring systems. In *Proceedings of the of the 13th International Conference on Artificial Intelligence in Education (AIED07)*, pages 701–702, Amsterdam, The Netherlands, 2007. IOS Press.

W. Stanley, R. Mathews, R. Buss, and S. Kotler-Cope. Insight without awareness: on the interaction of verbalization, instruction and practice in a simulated process control task. *Quarterly Journal of Experimental Psychology*, 41(A):553–577, 1989.

Albert L. Stevens and Allan Collins. The goal structure of a socratic tutor. In *Proceedings of the National ACM conference*, pages 256–263, New York, 1977. Association for computing Machinery.

Albert Stevens, Allan Collins, and Sarah E. Goldin. Misconceptions in students' understanding. In D. Sleeman and S. J. Brown, editors, *Intelligent Tutoring Systems*, Computers and Poeple, pages 13–24. Academic Press, 1982.

Rosemary J. Stevenson. Training quality and learning goals: Towards effective learning for all. *Behavioral and Brain Sciences*, 21(3):426–427, 1998.

Ron Sun, Todd Peterson, and Edward Merrill. Bottom-up skill learning in reactive sequential tasks. In *18th Cognitive Science Society Conference*, pages 684–690, Hillsdale, NJ, 1996.

Ron Sun, Edward Merrill, and Todd Peterson. From implicit skills to explicit knwoledge: A bottom-up model of skill learning. *Cognitive Science*, 25(2):203–244, 2001.

John Sweller and Paul Chandler. Evidence for cognitive load theory. *Cognition and Instruction*, 8(4):351–362, 1991.

J. Sweller and G.A. Cooper. The use of worked examples as a substitute for problem solving in algebra. *Cognition and Instruction*, 2(1):59–89, 1985.

J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12:257–285, 1988.

J. Sweller. Cognitive technology: Some procedures for facilitating learning and problem solving in mathematics and science. *Journal of Educational Psychology*, 81:457–466, 1989.

The OMEGA group. The nested mathematical theories implemented in ΩMEGA, url. http://www.ags.uni-sb.de/ mbase/content/omega/.

J. Trafton and B. Reiser. The contributions of studying examples and solving problems to skill acquisition, 1993.

D. R. Traum and J. F. Allen. Discourse obligations in dialogue processing. In *Proceedings 32nd Annual meeting of the Association for Computational Linguistics (ICSLP92)*, pages 1–8, 1994.

David R. Traum. Speech acts for dialogue agents. *Foundations of Rational Agency*, pages 169–201, 1999.

Dimitra Tsovaltzi and Armin Fiedler. An approach to facilitating reflection in a mathematics tutoring system. In *Proceedings of AIED Workshop on Learner Modelling for Reflection*, pages 278–287, Sydney, Australia, 2003.

Dimitra Tsovaltzi and Armin Fiedler. Enhancement and use of a mathematical ontology in a tutorial dialogue system. In *Proceedings of the IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 19–28, Acapulco, Mexico, 2003.

Dimitra Tsovaltzi and Armin Fiedler. Human-adaptive generation of natural language hints. In *International Joint Conference on Artificial Intelligence IJCAI05—5th Workshop on Computational Models of Natural Argument*, 2005.

Dimitra Tsovaltzi and Elena Karagjosova. A dialogue move taxonomy for tutorial dialogues. In Michael Strube and Candy Sidner, editors, *Proceedings of 5th SIGdial Workshop on Discourse and Dialogue*, Boston, USA, 2004.

Dimitra Tsovaltzi and Colin Matheson. Formalising hinting in tutorial dialogues. In *Proceedings of EDILOG: 6th Workshop on the Semantics and Pragmatics of Dialogue*, pages 185–192, Edinburgh, Scotland, UK, 2002.

Dimitra Tsovaltzi, Armin Fiedler, and Helmut Horacek. A multi-dimensional taxonomy for automating hinting. In James C. Lester, Rosa Maria Vicari, and Fábio Paraguaçu, editors, *Intelligent Tutoring Systems — 7th International Conference (ITS 2004)*, number 3220 in LNCS, pages 772–781. Springer, 2004.

Dimitra Tsovaltzi, Helmut Horacek, and Armin Fiedler. Building hint specifications in a NL tutorial system for mathematics. In *Proceedings of the 16th International Florida AI Research Society Conference (FLAIRS-04)*, Florida, USA, 2004.

Dimitra Tsovaltzi. Formalising hinting in tutorial dialogues. Master's thesis, The University of Edinburgh, Scotland, UK, 2001.

C. Ullrich, E. Andres, P. Krger, E. Melis, and M. Moormann. Tutorial component. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 2006.

K. VanLehn, R. M. Jones, and M. T. H. Chi. A model of the self-explanation effect. *Journal of the Learning Sciences*, 2(1):1–60, 1992.

Kurt VanLehn, Dumiszewe Bhembe, Min Chi, Colling Lynch, Kay Schulze, Robert Shelby, Linwood Taylor, Don Treacy, Andres Weinstein, and Mary Wintersgill. Implicit versus explicit learning of strategies in a non-procedural cognitive skill. In ITS2004 [ITS2004, 2004], pages 521–529.

Kurt VanLehn, Collin Lynch, Kay Schulze, Joel A. Shapiro, Robert Schelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, pages 147–204, 2005.

Kurt VanLehn. Cognitive skill acquisition. *Annual Review of Psychology*, 47:513–539, 1996.

Kurt VanLehn. The behaviour of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3):227–265, 2006.

B. Quoc Vo, C. Benzmüller, and S. Autexier. Assertion application in theorem proving and proof planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. poster description).

L. S. Vygotsky. *Mind in Society.* Harvard University Press, Cambridge, M.A., 1978.

Marilyn Walker and Rebecca Passonneau. Date: A dialogue act tagging scheme for evaluation of spoken dialogue systems, URL. `http://www.citeseer.nj.nec.com/walker01date.html`.

Hiroyoshi Watanabe, Kumiko Takai, Masayuki Arai, and Shigeo Takei. Case-based adviser for near-miss programs. In *Proceedings of AIED2003 Conference of Artificial Intelligence in Education*, pages 149–156, Sydney, 2003.

Ian Watson and Farhi Marir. Case-based reasoning: A review. *The Knowledge Engineering Review*, 9, 1994.

B. Weiner. *Human Motivation: metaphor, theories, and research.* Sage Publications Inc., 1992.

Sharon Alayne Widmayer. Schema theory: An introduction, URL. http://chd.gse.gmu.edu/immersion/knowledgebase.

Peter Wiemer-Hastings. The design and architecture of research methods tutor, a second generation dialog-based tutor. In *Proceedings of Dialog-based Intelligent Tutoring Systems: State of the Art and New Research Directions. Held in conjunction with ITS04, the Seventh International Conference on Intelligent Tutoring Systems*, pages 55–63, Maceiö, Brasil, 2004.

G.C. Williams and E.L. Deci. Internalization of biopsychosocial values by medical students: A test of self-determination theory. *Journal of Personality and Social Psychology*, 70:767–779, 1996.

G.C. Williams, Z.R. Freedman, and E.L. Deci. Supporting autonomy to motivate glucose control in patients with diabetes. *Diabetes Care*, 21:1644–1651, 1998.

D. Willingham, M. Nissen, and P. Bullemer. On the development of procedural knowledge. *Journal of Experimental Psychology*, 15:1047–1060, 1989.

Brent Wilson and Peggy Cole. A review of cognitive teaching models. *Educational Technology Research and Development*, 39 (4):46–64, 1991.

Lubwig Wittgenstein. *Tractatus Logico-Philosophicus.* Routledge Classics, London, 1975.

Magdalena Wolska, Bao Quoc Vo, Dimitra Tsovaltzi, Ivana Kruijff-Korbayová, Elena Karagjosova, Helmut Horacek, Malte Gabsdil, Armin Fiedler, and Christoph Benzmüller. An annotated corpus of tutorial dialogs on mathematical theorem proving. In *Proceedings of International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1007–1010, Lisbon, Portugal, 2004. ELDA.

David Wood and David Middleton. A study of assisted problem solving. *British Journal of Psychology*, 66:181–191, 1975.

D. J. Wood, J. S. Bruner, and G. Ross. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17:89–100, 1976.

H. Wu. The mathematician and the mathematics education reform. In Notices of the American Mathematical Society, December 1996.

H. Wu. What is so difficult about the preparation of mathematics teachers. In National Summit on the Mathematical Education of Teachers: Meeting the Demand for High Quality Mathematics Education in America, November 2001.

Zhou Yujian, Reva Freedman, Michael Glass, Joel A. Michael, Allen A. Rovick, and Martha W. Evens. What should the tutor do when the student cannot answer a question? In AAAI Press, editor, *Proceedings of the Twelfth International Florida AI Research Society Conference (FLAIRS-99)*, pages 187–191, Orlando, FL, 1999.

Juan-Diego Zapata-Rivera and Jim E. Greer. Analysing student reflection in *The Learning Game. In* Proceedings of AIED Workshop on Learner Modelling for Reflection*, pages 288–297, Sydney, Australia, 2003.*

*Yujian Zhou, Reva Freedman, Michael Glass, Joel A. Michael, Allen A. Rovick, and Martha W.. Evens. Delivering hints in a dialogue-based intelligent tutoring system. In AAAI Press and MIT Press, editors,* Proceedings Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 128–134, Orlando FL, 1999. Published in one volume with Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI-99).*

*Claus Zinn, Johanna D. Moore, Mark G. Core, Sebastian Varges, and Kaśka Porayska-Pomsta. The be&e tutorial learning environment (beetle). In* Proceedings of Diabruck, the 7th Workshop on the Semantics and Pragmatics of Dialogue*, Saarbrücken, Germany, 2003.*

*Claus Zinn. A 3-tier planning architecture for managing tutorial dialogue. In S. A. Cerri, G. Gouardéres, and F. Paraguaçu, editors,* Proceedings of Intelligent Tutoring Systems, Sixth International Conference*, Biarritz, France, 2002.*

# Index