A Human Oriented Proof Presentation Method

Huang Xiaorong

SEKI Report SR-89-11

A Human Oriented Proof Presentation Method

Huang Xiaorong
Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049, D-6750 Kaiserslautern, W.-Germany

This research was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D2)

A Human Oriented Proof Presentation Model

Xiaorong Huang

Fachbereich Informatik, Universität Kaiserslautern

Kaiserslautern, West Germany

E-mail: huang@uklirb.uucp

Abstract

It has long become a general consensus in the field of natural language generation that the whole generation task can be divided into two relatively independent parts: a so called text planner that decides on "what to say" and a generator that decides on "how to say it". This paper describes a text planner that decides on "what to say", currently under development at the University of Kaiserslautern for our automated theorem prover MKRP. The most important observation of this paper is, that the transformation task of this planner is primarily neither a *linguistic* problem nor a *purely logical* problem in the traditional sense: the transformation steps presented in this paper are based on a *mental model of human proof presentation* and our main effort was to make this model cognitively adequate. The model of human presentation consists primarily of the following three parts :

1. The identification of the size of human mathematical reasoning steps. The main observation is that human style inference steps usually have the "size" of at least an application of an axiom or a theorem. An algorithm is developed to generate domain-specific compound inference rules from axioms and theorems, which are then used to raise proofs to the so called *conceptual level* .

2. The ordering or "linearization" of proof parts: besides the logical constraint that proof lines must be first proved before they can be used, other pragmatic constraints on the ordering used in the "natural" human proof presentation are identified. In addition, a focus mechanism developed for natural language discourse is adapted and integrated to provide a total order.

3. Reference choices for each step: after the ordering each proof step will be translated into one message unit, which corresponds roughly to a sentence in the output text. Therefore, for each step we must decide how to refer to the used inference rules and the proof lines used as premises for this step, which parts to mention explicitly, implicitly and which ones to omit. A proof unit model is designed to attack this problem.

Content

1. Introduction

It has long become a general consensus in the field of natural language generation that the whole generation task can be divided into two relatively independent parts: a so called text planner that decides on "what to say" [McKeown 85] and a generator that decides on "how to say it" [McDonald 83]. Roughly speaking, the planner accepts information from the underlying application program and then selects a subset, which is relevant for the current discourse purpose. It should then organize the chosen information into an appropriate text form and produce a sequence of ordered paragraphs and sentences in some internal code. This is the so called message which is now taken over by a generator or tactical component. It transforms the internal code into natural language using a dictionary that contains natural language interpretations for tokens in the internal knowledge representation language. This paper describes such a text planner we are developing at the University of Kaiserslautern for our automated theorem prover MKRP [MKRP 84]. See Fig 1 for an overview of the whole system architecture. In our case the theorem prover is the underlying application program and the information the planner accepts will be a Gentzen style natural deduction proof. Output of the planner will be a sequence of ordered internal code units each of which corresponds roughly to a sentence in the final natural language output. This text planner together with a natural language generator constitutes the Logic-to-Natural-Language Translator in Fig. 1.

The most important observation of this paper is, that the transformation task of this planner is primarily neither a *linguistic* problem nor a *purely logical* problem in the traditional sense. This observation changed the course of our original work plan: the transformation steps presented in this paper are based on a *mental model of human proof presentation* and our main effort was to make this model cognitively adequate.

After this transformation at the conceptual level, the final transformation into natural language is then (we believe) more or less straightforward. Although this model is entirely logic oriented, striking similarities can be found between this model and the theories and methods developed for the natural language discourse in recent years. The model of human presentation consists primarily of the following three parts:

1. The identification of the size of human mathematical reasoning steps. In particular, it will be argued that the size of reasoning steps of a Gentzen proof is normally much too small. Then we will show how domain-specific inference rules can be automaticlly generated that are used to raise the size of reasoning steps.

2. The ordering or "linearization" of proof parts and proof lines: besides the logical constraint that proof lines must be first proved before they can be used, other pragmatic constraints on the ordering used in the "natural" human proof presentation are identified. In addition, a focus mechanism developed for natural language discourse is adapted and integrated to provide a total order.

3. Reference choices for each step: after the ordering each proof step will be translated into one message unit. Therefore, for each step we must decide how to refer to the used inference rules and

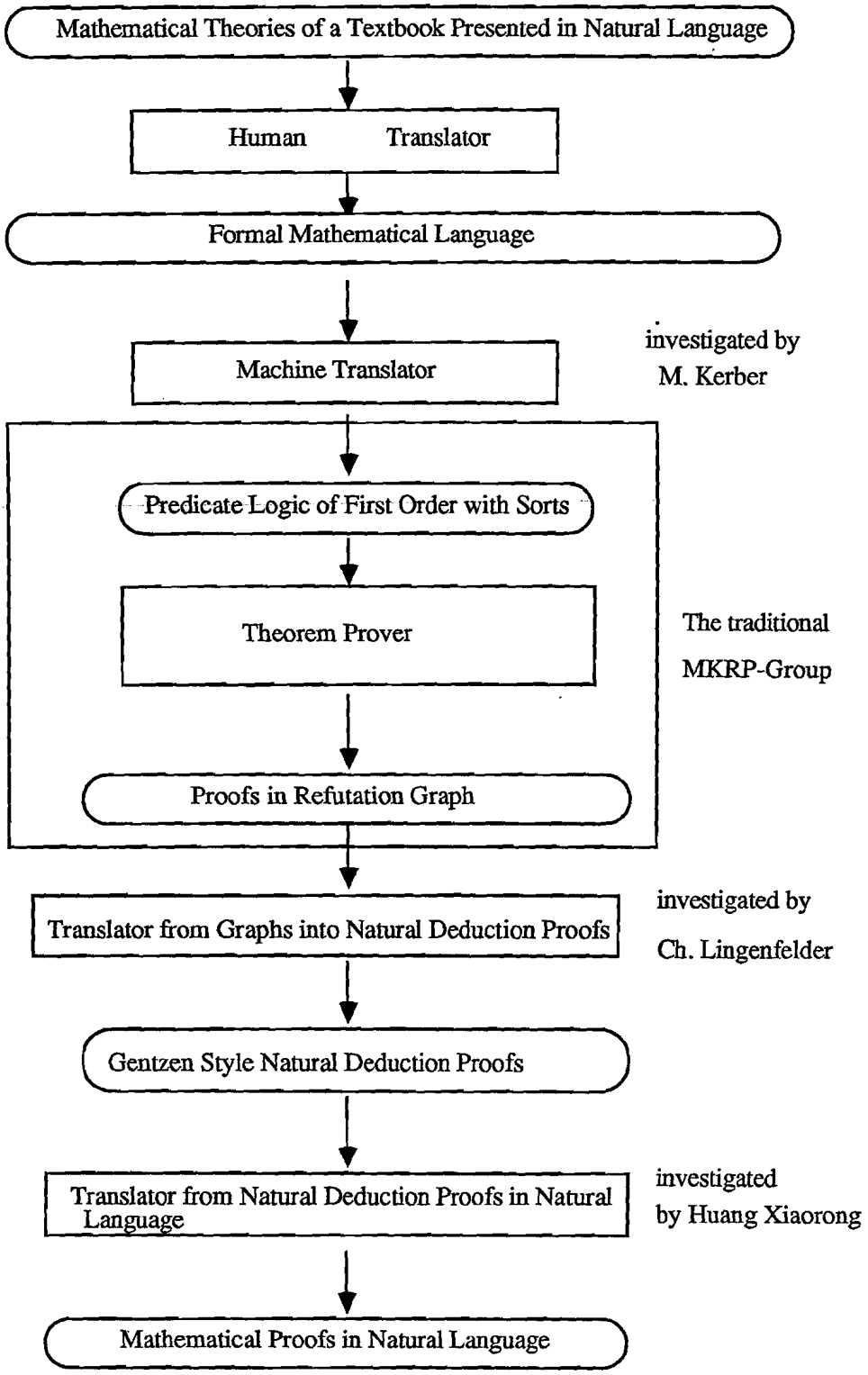

Fig. 1

the proof lines that have already been proved in the previous context and are used as premises for this step. We may decide to refer to them explicitly, which leads to translations such as "because 1 is the unit of F, by the definition of unit, ..." . On the contrary, we may also omit them, for examle,

if a reason has been proved just before or if an inference rule is already familiar to the user. This may lead to a translation as simple as "thus" or "therefore". A proof unit model is designed to attack this problem.

In the rest of this section we give a short introduction to the Gentzen calculus, which is the formalism of our input proofs, and a short introduction to the three components of the model. The three main components of our proof presentation model and the transformations will be dealt with in more detail in section 2, section 3, and section 4, respectively. Section 5 presents our plans for future work, including the connection to the tactical component. A complete example is used through the whole paper, which shows how an input proof of forty-seven proof lines is translated into a sequence of fourteen message units.

The calculus developed by G. Gentzen in 1933 [Gentzen 1935] is called "natural" because its structural inference rules resemble the normal inference manner of human mathematicians: for example, assumptions are first introduced and then discharged after conclusions are drawn, making the conclusion independent of the assumptions; proofs are divided into cases; and so on. Yet the examples in Section 2 reveal that not enough is done to increase the "inference step size". While, for example, a human mathematician will derive $a \in F$ from the facts $a \in U$ and $U \subseteq F$ in one step, eight Gentzen proof lines are needed to draw the same conclusion. Small and tedious proof steps make even simple proofs long and difficult to understand. In section 2 we will argue that human mathematicians usually reason at a higher level which we call the "conceptual" level. We first try to pin down the size of the proof steps at the conceptual level and claim that they should correspond to one application of either an axiom, a definition, a proved theorem or a lemma. Then we show formally how a proof step of this "size" can be achieved by rules which are "compound" Gentzen inference rules. The problem is that usually only some basic and general rules are included in a calculus, as there is no universal way to incorporate all those infinitely many compound rules. As the first important observation of this paper, however, we claim that if we concentrate on a specific mathematical domain, there is always a rather small set of compound inference rules each corresponding to an application of an axiom or a theorem of this particular domain which are in fact the only compound rules used and can thus be picked out and integrated into the Gentzen calculus. An algorithm is designed in Section 2 that generates these domain-specific compound inference rules from the axioms and theorems of a mathematical theory. These newly generated rules, together with other standard Gentzen rules, essentially form a new domain-specific calculus. A fairly simple algorithm is then used to transform the original Gentzen proof into a proof in this new calculus, where only those newly generated domain-specific rules and the structural Gentzen rules are used. The following is a brief introduction to the Gentzen calculus, the generation of domain-specific rules is then discussed in section 2.

The calculus developed by G. Gentzen differs from earlier axiomatic systems in that it provides a mechanism for introducing and discharging assumptions. This means that new logical conclusions are derived from assumptions introduced rather than from logical axioms which are part of the calculus. The following is the Gentzen style calculus we are using at present. "D" and "I" mean "deletion" and "introduction", respectively. $F \vdash G$ means G is derivable from F in the Gentzen calculus. For convenience of presentation, inference rules are characterized as structural and non-structural: Structural rules are those helpful for determining the overall structure of a proof: introducing or discharging assumptions, dividing into cases, etc. Here $\perp$ denotes the contradiction.

Structural Gentzen Rules:

$$\frac{}{F \vdash F} \qquad \frac{F \vdash G}{F \Rightarrow G} \qquad \frac{F \vee G,\ F \vdash H,\ G \vdash H}{H} \qquad \frac{\exists x\, F(x),\ F(a) \vdash H}{H}$$

HYPothesis **DEDuction** **CASE** **CHOICE**

$$\frac{F, \neg G \vdash \bot}{F \vdash G}$$

IP(Indirect Proof)

Non-Structural Gentzen Rules

$$\frac{F,\ G}{F \wedge G} \qquad \frac{F}{F \vee G} \quad \frac{G}{F \vee G} \qquad \frac{F(a)}{\forall x\, F(x)} \qquad \frac{F(a)}{\exists x\, F(x)}$$

$\wedge$**I** $\vee$**I** $\forall$**I** $\exists$**I**

$$\frac{F \wedge G}{F} \quad \frac{F \wedge G}{G} \qquad \frac{F,\ F \Rightarrow G}{G} \qquad \frac{\forall x\, F(x)}{F(a)} \qquad \frac{F,\ \neg F}{\bot} \qquad \frac{F \Leftrightarrow G}{F \Rightarrow G} \quad \frac{F \Leftrightarrow G}{G \Rightarrow F}$$

$\wedge$**D** $\Rightarrow$**D** $\forall$**D** $\neg$**D** $\Leftrightarrow$**D**

Every figure above shows an inference rule. Formula schemes separated by comma above the bar represent premises. Premises are also called *reasons*. Now a Gentzen proof is a sequence of proof lines each of the form:

line-No assumption-set $\vdash$ a derived formula Inference-rule-name(reason-pointers)

where assumption-set is a set of proof lines introduced as assumptions this particular line depends on. Reason-pointers, on the other hand, is a set of proof lines used by the inference rule in deriving this new line. The last line of a proof is the conclusion line, whose assumption set is either empty or contains only assumptions, which are actually the premises to the theorem to be proved. The derived formula is often called a proof line too as far as no ambiguity occurs. The following is a very simple Gentzen proof:

1.	1	$\vdash$	A	HYP
2.	2	$\vdash$	B	HYP
3.	1,2	$\vdash$	A$\wedge$B	$\wedge$(1,2)

Here line 1 and 2 introduce two assumptions A and B. The "and introduction rule" is used in drawing the third line which is therefore dependent on the first two lines.

After the input proof is raised to the conceptual level by the domain-specific inference rules, we shall enforce an appropriate order on proof parts and proof lines. Instead of a direct linearization, however, we first try to build an ordered proof tree. This tree structure maintains the complete information about the proof structure which will be used later on. A post-order traversal will provide a linearization. The nodes of the proof trees are proof lines, and the links represent the

reason-logical-successor relation. After constructing an initial tree from the raised input proof, which is the output of the first component, the task is now to enforce an order on the children of each node. The following three ordering criteria are identifed: first of all, the logical constraint says that reason must precede logical successors. This is in fact guaranteed by the tree structure. Secondly, we claim that many structural Gentzen inference rules entail a particular order in which its reasons must be presented. Finally, if these two contraints do not suffice to enforce a total order, a focus mechanism is tried. The basic principle of this mechanism is similar to the focus mechanism developed for natural language discourse: In the case of mathematical proofs, it says once we start to prove properties of a particular object (collection), which is the focus space at the moment, we will first try to continue in this space before we turn to another focus space, as far as this is allowed by the other two constraints.

The main task of the final step of our model for proof presentation is to generate one message unit from every node of the odered proof tree, while the proof process traverses the tree in post-order. At each node, which is a proof step, the decision about "what to say"must be made. For both inference rules and reasons, three reference forms are identified: explicit, implicit, and "omit". For example reasons proved far before in the context may be repeated explicitly, while reasons just proved will usually be omitted. In other cases, an implicit hint might be chosen. A very important observation is that the choices on reference forms for reasons depend on the structural relationship between the proof unit where the reasons are proved on the one hand, and the current active proof unit on the other hand, as well as the "physical" distance between the point where the reasons last appear and the point where they are used. A proof unit model that identifies proof units and specifies relationships between proof units is described in section 4. It will be also argued that as opposed to the references for reasons, the reference choice for inference rules are not context sensitive.

2. Domain-Specific Compound Inference Rules

We want to determine the size of the inference steps that human mathematicians usually prefer. While it is certainly true that mathematicians reason at very different levels of abstraction depending on their background, skill and also on their field of expertise, we are nevertheless arguing that they all reason on the level of mathematical concepts,which are connected by axioms and later by proved theorems. As an example we have taken the very first theorem of an introductory book on algebra and automata [Deussen 71] and encoded it. The result is a long proof with many small "logical level" (instead of conceptual level) steps. In this section, a simple algorithm for deriving conceptual level inference rules will be discussed, it will be shown, how these rules can be used to reorganize a proof, and how to abstract from the "logical level" to the conceptual level. While these two terms are not very precise and the boarder line is somewhat blurred, it will nevertheless become clearer on how much we can simplify a proof using these ideas.

Let us first have a look at the example and explain our ideas in more detail. Suppose we have the concept of "subset" encoded in the following way:

$$U \subseteq F \Leftrightarrow \forall x \; x \in U \Rightarrow x \in F$$

Now everyone with a standard mathematical training will find it natural to deduce $a \in F$ given the fact that $U \subseteq F$ and $a \in U$. What we want to say is, even if he is not familiar with set theory, he will immediately reason at this level of abstraction, just as if he were using a calculus with an inference rule like:

$$\frac{a1 \in U1,\ U1 \subseteq F1}{a1 \in F1} \qquad \text{where a1, U1, F1 denote arbitrary terms} \qquad 2.1$$

This single inference step at the *conceptual level* corresponds to the following Gentzen-proof segment, i.e. the following steps at the *logical level*:

1.	1	$\vdash$	$U1 \subseteq F1$	HYP
2.	2	$\vdash$	$a1 \in U1$	HYP
3.	3	$\vdash$	$\forall F,U\ U \subseteq F \Leftrightarrow \forall x\ x \in U \Rightarrow x \in F$	HYP
4.	3	$\vdash$	$U1 \subseteq F1 \Leftrightarrow \forall x\ x \in U1 \Rightarrow x \in F1$	$\forall$D(3)
5.	3	$\vdash$	$U1 \subseteq F1 \Rightarrow \forall x\ x \in U1 \Rightarrow x \in F1$	$\Leftrightarrow$D(4)
6.	1, 3	$\vdash$	$\forall x\ x \in U1 \Rightarrow x \in F1$	$\Rightarrow$D(1, 5)
7.	1, 3	$\vdash$	$a1 \in U1 \Rightarrow a1 \in F1$	$\forall$D(6)
8.	1, 2, 3	$\vdash$	$a1 \in F1$	$\Rightarrow$D(2, 7)

Eight Gentzen proof steps are used: three to introduce the given facts and the set inclusion definition, two to instantiate the universal quantifier, one to split equivalence, and two applications of Modus Ponens.The point is that the conciseness at the conceptual level is not really based on the person's competence in this particular area, and the tediousness at the logical level is not caused by the machine's inability of finding elegant proofs. The reasoning simply takes place at different levels of abstraction.

Rules like 2.1 are domain-specific, because they are only defined for domain-specific predicates (in our case $\subseteq$ and $\in$, both written in infix form in our example). Yet it is not difficult to find their general counterpart. For example, the general counterpart of inference rule 2.1 is given as 2.2.

$$\frac{P(x\ y) \Leftrightarrow \forall z\ (Q(z\ x) \Rightarrow Q(z\ y)),\ P(a\ b),\ Q(c\ a)}{Q(c\ b)} \qquad \text{where a, b, c denote arbitrary terms} \qquad 2.2$$

It is straightforward to prove that 2.2 is a compound rule in the sense that it can be taken as an abbreviation of a sequence of basic Gentzen rules and is therefore correct. Why then did Gentzen call his calculus "natural deduction" and what makes our compound rules more natural? The point is, that Gentzen designed his calculus for any mathematical field (i.e. not domain-specific) on the observation that human mathematicians usually develop proofs in such a way: they introduce assumptions (HYPothesis Rule) and later discharge them, i.e. make the conclusion independent of the assumptions (DEDuction Rule); they prove by cases (Rule of CASE), and so on. This means that the "structural" Gentzen inference rules provide a "structural" means to construct and develop proofs similar to the way human mathematicians are used to. But its naturalness is therefore also limited. We might call it "structurally natural" in fact. That means that those non-structural infererence rules are still basic and usually too small as an appropriate "inference step size". While it is possible to introduce the sort of particular compound rules as mentioned above (2.2), there is practically no way to do that in general, for generally no specific compound inference rules have dominance over others and there are infinitely many of them.

If we concentrate on a special area of mathematics, however, there are rules such as 2.1 or 2.2 that are distinguished from others and worth separating. Indeed, they usually correspond exactly to one application of either an axiom or a theorem. A quite simple but powerful method is developed below to derive domain-specific compound inference rules from axioms and theorems. These compound inference rules, affected by the specific encoding manner of the axioms or theorems, reflect a certain degree of subjective reasoning style, as opposed to the general "objective" rules of G. Gentzen. New rules are then incrementally added to the calculus as new theorems or even intermediate results or lemmas are proved, which corresponds to the observation that in a mathematical textbook the reasoning usually starts at some very basic level and becomes more and more "abstract" and conceptual towards the end, where the kowledge of the reader is assumed to contain all the previously introduced material. We will show that astonishingly all the conceptual level inference rules needed in our examples can be derived by this quite simple method. The resulting level of abstraction turns out to be quite natural and satisfactory. The basic observation is that mathematicians apply axioms and theorems as a fundamental unit of reasoning and skip those small logical manipulations enforced at the Gentzen calculus level. The table 2.1 summarizes our rules that can be used to derive compond inference rules.

Rule 2 in table 2.1 reflects the basic idea, that axioms and theorems are usually given in implication form. Taking into account that implications can be nested, Rule 3 is designed to make a recursion out of Rule 2 (and of course also other rules). Rule 4 is a recursion in the other direction. Rule 5 splits the equivalences that often appears in axioms and theorems in order to apply rule 2. Rule 6 to Rule 8 save the effort to split or combine disjunctive and conjunctive formulas. (Therefore, a slight change is made to the syntax, that is to apply a rule, it suffices to find for each reason P a preceding proof line in the form of ...$\wedge$ P $\wedge$...., instead of a proof line in form P, as required in the Gentzen calculus. Rule 1 is used to introduce axioms and theorems that other rules may start with.

Let us have a look again at the above subset example and see how the axiom can be used to derive inference rules. Notice the propagation of quantifiers is not given explicitly in our discussion. The axiom is:

$$U \subseteq F \Leftrightarrow \forall x \; x \in U \Rightarrow x \in F$$

As the first step we always apply Rule 1. Then in our case we apply Rule 5 to split the equivalence. Now we proceed from one of the two new inference rules produced by Rule 5:

$$\frac{}{\forall F,U \; U \subseteq F \Rightarrow \forall x \; x \in U \Rightarrow x \in F} \qquad 2.3$$

According to Rule 3, Rule 2 can be applied repeatedly and produces

$$\frac{U1 \subseteq F1}{\forall x \; x \in U1 \Rightarrow x \in F1} \qquad 2.4$$

and

$$\frac{\neg (\forall x \; x \in U1 \Rightarrow x \in F1)}{\neg (U1 \subseteq F1)} \qquad 2.5$$

	RULES FOR DERIVING NEW INFERENCE RULES	
	Given Facts and inference Rules	New Special Purpose Inference Rule
1.	P, given as axiom or theorem	$\frac{}{P}$
2.	$\forall x\ P(x) \Rightarrow Q(x)$	$\frac{P(a)}{Q(a)}$ and $\frac{\neg Q(a)}{\neg P(a)}$ where a denotes an arbitrary term
3.	$\frac{P}{Q}$ and $\frac{P'}{R'}$ is derivable[1] from Q	$\frac{P, P'}{R'}$
4.	$\frac{P}{Q}$, $\frac{P'}{P1}$ and $P1 \vdash P$	$\frac{P'}{Q}$
5.	$\frac{P}{\forall x\ Q1 \Leftrightarrow Q2}$	$\frac{P}{\forall x\ Q1 \Rightarrow Q2}$ and $\frac{P}{\forall x\ Q2 \Rightarrow Q1}$
6.	$\frac{P1 \vee P2}{Q}$	$\frac{P1}{Q}$ and $\frac{P2}{Q}$
7.	$\frac{P1 \wedge P2}{Q}$	$\frac{P1, P2}{Q}$
8.	$\frac{P}{Q1 \wedge Q2}$	$\frac{P}{Q1}$ and $\frac{P}{Q2}$

Table 2.1

Applying Rule 3 and Rule 2 on 2.4 once more, we again obtain two new rules:

$$\frac{a1 \in U1,\ U1 \subseteq F1}{a1 \in F1} \qquad 2.6$$

$$\frac{a1 \notin F1,\ U1 \subseteq F1}{a1 \notin U1} \qquad 2.7$$

where 2.6 is in fact 2.1. Because $\neg\ (\forall x\ x \in U1 \Rightarrow x \in F1)$ can be rewritten into $\exists x\ x \in U1 \wedge x \notin F1$, combined with the $\exists I$ Rule of the Gentzen calculus, we can apply Rule 4 on 2.5 to produce 2.8:

$$\frac{a \in U1,\ a \notin F1}{\neg(U1 \subseteq F1)} \qquad 2.8$$

Notice that Rule 4 is the only one that can not be carried out in a fully automatic way. But we believe it is easy to develop some simple heuristic methods to cover the common rewriting cases, for example that negation should often be propagated inside implication and rewritings between pairs $\neg\forall$ and $\exists$, and $\neg\exists$ and $\forall$. The following is the example which we will use through the paper. All compound inference rules used in our example are listed. They are derived from the five

[1] It means inference rule $\frac{P'}{R'}$ is derivable by this algorithm from formula Q.

definitions and one lemma which is supposed to be proved in advance and used in the proof of the theorem. The original proof of forty-seven proof lines is reduced to a proof of fifteen lines.

Theorem: 1.9 Satz 1.9 in [Deussen 71]

group(F op) $\wedge$ subgroup(U F op) $\wedge$ unit(F 1 op) $\wedge$ unit(U 1_u op) $\Rightarrow$ 1= 1u

If F is a group, U is a subgroup of G, 1, 1_u the unit element of F and U respectively, then 1= 1u

Definitions, axioms and lemmas as context

t1. Def.

$\forall$ F ,OP semigroup(F OP) $\Leftrightarrow \forall x,y,z\ x\in F \wedge y\in F \wedge z\in F \Rightarrow (xy)z=x(yz)$

$\wedge (\forall x,y\ x\in F \wedge y\in F \Rightarrow xy\in F)$ (not used in this example)

t 2. Def.

$\forall 1,F,OP$ unit(F 1 OP)$\Leftrightarrow$ semigroup(F OP) $\wedge\ 1\in F \wedge (\forall f\ f\in F \Rightarrow 1f=f1=f)$

(Note 1f=f1=f is an abreviation of 1f=f $\wedge$ f1=f, first order predicate logic with equality is assumed. And expressions in form of apply(op x y) are abreviated to xy throughout our example, for simplicity.)

t 3. Def.

$\forall F,OP$ group(F OP)$\Leftrightarrow$semigroup(F OP)$\wedge(\exists 1$ unit(F 1 OP) $\wedge$

$(\forall f\ f\in F \Rightarrow \exists f^{-1}\ f^{-1}\in F \wedge f^{-1}f=1)$

t 4. Def.

$\forall f,g,x,F,OP$ solution(f g x F OP)$\Leftrightarrow$ semigroup(F OP) $\wedge$ f,g,x$\in$ F $\wedge$ fx=g

(f,g,x$\in$ F is an abreviation of f$\in$ F $\wedge$ g$\in$ F $\wedge$ x$\in$ F)

t 5. Lemma

$\forall f,g,x_1,x_2,F,OP$ group(F OP) $\wedge$ solution(f g x_1 F OP)$\wedge$solution(f g x_2 F OP)$\Rightarrow x_1=x_2$

t 6. Def

$\forall$ U,F,OP subgroup(U F OP)$\Leftrightarrow$semigroup(F OP) $\wedge$ U$\subseteq$F $\wedge$ group(U OP)

t7. Def.

$\forall$ U,F U$\subseteq$F$\Leftrightarrow \forall x\ x\in U \Rightarrow x\in F$

Proof 1: Input proof in Gentzen Calculus

F, U, OP, 1, 1_u are constants. Notice a slight syntactic change has been made: in the assumption-set part of proof lines, line numbers of proof lines introduced by the HYP rule are preceded by a symbol "a", standing for assumption.

1.	a1	⊢	group(F op) ∧ subgroup(U F op) ∧ unit(F 1 op) ∧ unit(U 1_u op)	HYP
2.	a1	⊢	group(F op)	∧D(1)
3.	a1	⊢	subgroup(U F op)	∧D(1)
4.	a1	⊢	unit(F 1 op)	∧D(1)
5.	a1	⊢	unit(U 1_u op)	∧D(1)
6.	t2	⊢	unit(U 1_u op)⇔ semigroup(U op) ∧ 1_u ∈ U ∧ (∀f f∈ U⇒1_uf=f1_u=f)	∀D(t2)
7.	t6	⊢	subgroup(U F op)⇔semigroup(F op) ∧ U⊆F ∧ group(U op)	∀D(t6)
8.	t6	⊢	subgroup(U F op)⇒semigroup(F op) ∧ U⊆F ∧ group(U op)	⇔D(7)
9.	a1,t6	⊢	semigroup(F op) ∧ U⊆F ∧ group(U op)	⇒D(3,8)
10.	a1,t6	⊢	U⊆F	∧D(9)
11.	t2	⊢	unit(U 1_u op)⇒ semigroup(U op) ∧ 1_u ∈ U ∧ (∀f f∈ U⇒1_uf=f1_u=f)	⇔D(6)
12.	t2, a1	⊢	semigroup(U op) ∧ 1_u ∈ U ∧ (∀f f∈ U⇒1_uf=f1_u=f)	⇒D(5,11)
13.	t2, a1	⊢	1_u ∈ U	∧D(12)
14.	t2,a1	⊢	∃x x∈ U	∃I(13)
15.	a15	⊢	u∈ U	HYP
16.	t2, a1	⊢	∀f f∈ U⇒1_uf=f1_u=f	∧D(12)
17.	t2, a1	⊢	u∈ U⇒1_uu=u1_u=u	∀D(16)
18.	t2, a1,a14	⊢	1_uu=u1_u=u	⇒D(15,17)
19.	t2, a1,a15	⊢	u1_u=u	∧D(18)
20.	t7	⊢	U⊆F⇔∀x x∈ U ⇒ x∈ F	∀D(t7)
21.	t7	⊢	U⊆F⇒∀x x∈ U ⇒ x∈ F	⇔D(20)
22.	a1,t6,t7	⊢	∀x x∈ U ⇒ x∈ F	⇒D(10,21)
23.	a1,t6,t7	⊢	u∈ U ⇒ u∈ F	∀D(22)
24.	a1,a15,t6,t7	⊢	u∈ F	⇒D(15,23)

25.	a1,t6,t7	⊢	$1_u \in U \Rightarrow 1_u \in F$	∀D(22)
26.	a1,t2,t6,t7	⊢	$1_u \in F$	⇒D(13,25)
27.	t6, a1	⊢	semigroup(F op)	∧D(9)
28.	a1,a15,t2,t6,t7	⊢	semigroup(F op) ∧ u∈F ∧ $1_u \in F \wedge u1_u = u$	∧I(24,26,27,19)
29.	t4	⊢	solution(u u 1_u F op)⇔ semigroup(F op)	
			∧ u, 1_u,u∈F ∧ $u1_u$=u	∀D(t4)
30.	t4	⊢	semigroup(F op) ∧ u, 1_u,u∈F ∧ $u1_u$=u	
			⇒solution(u u 1_u F op)	⇔D(29)
31.	a1,a14,t2,t4,t6,t7			
		⊢	solution(u u 1_u F op)	⇒D(28,30)
32.	t2	⊢	unit(F 1 op)⇔ semigroup(F op) ∧ 1 ∈F	
			∧ (∀f f∈F⇒1f=f1=f))	∀D(t2)
33.	t2	⊢	unit(F 1 op)⇒ semigroup(F op) ∧ 1 ∈F	
			∧ (∀f f∈F⇒1f=f1=f))	⇔D(32)
34.	a1,t2	⊢	semigroup(F op) ∧ 1 ∈F ∧ (∀f f∈F⇒1f=f1=f)	⇒D(4,33)
35.	a1,t2	⊢	∀f f∈F⇒1f=f1=f	∧D(34)
36.	a1,t2	⊢	u∈F⇒1_u=u1=u	∀D(35)
37.	a1,a14,t6,t7,t2	⊢	1_u=u1=u	⇒D(24,36)
38.	a1,a14,t6,t7,t2	⊢	u1=u	∧D(37)
39.	t2	⊢	1∈F	∧D(34)
40.	a1,a14,t6,t7,t2	⊢	semigroup(F op) ∧ u∈F ∧ 1 ∈F ∧ u∈F ∧ u1=u	∧I(27,38,39,24)
41.	t4	⊢	solution(u u 1 F OP)⇔ semigroup(F op)	
			∧ u, 1,u∈F ∧ u1= u	∀D(t4)
42.	t4	⊢	semigroup(F op) ∧ u,1,u∈F ∧ u1= u	
			⇒solution(u u 1 F op)	⇔D(41)
43.	a1,a14, t4 ,t6,t7,t2	⊢	solution(u u 1 F op)	⇒D(40,42)
44.	a1,a14, t4 ,t6,t7,t2	⊢	group(F op) ∧ solution(u u 1 F op)	

			$\wedge$ solution(u u 1_u F op)	$\wedge$I(2,31,43)
45.	t5	$\vdash$	group(F op) $\wedge$ solution(u u 1 F op)	
			$\wedge$solution(u u 1_u F op)$\Rightarrow$1= 1_u	$\forall$D(t5)
46.	a1,a14, t4 ,t6,t5,t7,t2			
		$\vdash$	1= 1_u	$\Rightarrow$D(44,45)
47.	a1,t4 ,t6,t5,t7,t2	$\vdash$	1= 1_u	Choice(46,14)

Compound Inference Rules generated from axioms and lemmas actually used in the proof

For space limitation only rules used in our example proof are listed. For example, t1.2 is a rule derived from axiom t1. The symbols F1, U1, OP1, 1, u1 can be substituted by any well formed terms.

t2.1 $\dfrac{\text{unit}(F_1\ 1\ OP_1)}{1 \in F_1}$ t2.2 $\dfrac{\text{unit}(F_1\ 1\ OP_1),\ u_1 \in F_1}{u_1 1 = u_1}$

t3.1 $\dfrac{\text{group}(F_1\ OP_1)}{\text{semigroup}(F_1\ OP_1)}$

t4.1 $\dfrac{a_1, b_1, c_1 \in F_1,\ a_1c_1=b_1,\ \text{semigroup}(F_1\ OP_1)}{\text{solution}(a_1\ b_1\ c_1\ F_1\ OP_1)}$

t5.1 $\dfrac{\text{solution}(a_1\ b_1{}'\ c_1\ F_1\ OP_1),\ \text{solution}(a_1\ b_1\ c_1\ F_1\ OP_1),\ \text{group}(F_1\ OP_1)}{b_1 = b_1}$

t6.1 $\dfrac{\text{subgroup}(U_1\ F_1\ OP_1)}{U_1 \subseteq F_1}$

t7.1 $\dfrac{U_1 \subseteq F_1,\ a_1 \in U_1}{a_1 \in F_1}$

Proof two: reduced by applying compound inference rules above

OP, 1, 1_u are constants. They have nothing to do with the symbols used in inference rules.

1.	a1	$\vdash$	group(F OP) $\wedge$ subgroup(U F OP) $\wedge$ unit(F 1 OP) $\wedge$ unit(U 1_u OP)	hyp
2.	a1.2	$\vdash$	U$\subseteq$F	t6.1(1.2)
3.	a1.4	$\vdash$	$1_u \in$ U	t2.1(1.4)
4.	a1.4	$\vdash$	$\exists$x x$\in$U	$\exists$I(3)
5.	a5	$\vdash$	u$\in$U	hyp
6.	a1.4,a2	$\vdash$	u1_u=u	t2.2(1.4, 5)
7.	a1.2,a2	$\vdash$	u$\in$F	t7.1(2, 5)
8.	a1.2, a1.4	$\vdash$	$1_u \in$F	t7.1(2, 3)
9.	a1.1	$\vdash$	semigroup(F OP)	t3.1(1.1)
10.	a1.1, a1.2, a1.4,a2			

		⊢	solution(u u 1_uF OP)	t4.1(6,7,8,9)
11.	a1.2, a1.3,a2	⊢	u1=u	t2.2(1.3,7)
12.	a1.3	⊢	1 ∈ F	t2.1(1.3)
13.	a1.1,a1.2,a1.3, a2			
		⊢	solution(u u 1 F OP)	t4.1(7,11,12,9)
14.	a1, a2	⊢	1= 1u	t5.1(1.1,10,13)
15.	a1	⊢	1= 1u	Choice(4,14)

Notice the assumption set of proof lines. For example a1.2 indicates the second conjunctive subexpression of the line a1. In the last two lines, it is simply written as a1, for all subexpressions are included.

QED.(Satz 1.9)

The reader may have noticed that axioms and theorems do not appear in the proof as premises (proof line in fact) any more, once they are used to guide the derivation of new compound inference rules. But the "deductive ability" of the overall system is not weakened since axioms and theorems are provable with these compound inference rules and general Gentzen rules (A formal proof is omitted). Indeed, it is not completeness and correctness problems we are primarily concerned with, but the naturalness of these derived rules, which we believe corresponds to the naturalness of the given axioms and theorems (or rather their encoding), and closely related, it also depends on the depth of the recursive application of Rule 2. Of course, more careful studies are necessary. Furthermore, derivation rules for encoding patterns other than the implication (Rule 2 in our algorithm) might be useful too. In order to produce really natural proofs, structural information of the Gentzen proof must be used such that more compound inference rules will be generated not only from given lemmas or previously proved theorems, but also from intermediate results from within a proof.

The next problem, after the derivation of domain-specific compound rules, is how to shorten the original Gentzen proof. A brute force method is currently used in our implimentation. We simply test first for each proof line if there is an applicable compound rule in a bottom-up direction (since rules are all disjoint there is no problem of conflict and no special treatment for the resolution of conflicts is needed right now. This may be necessary, however, once we have more rules, particularly if they are non-disjoint). If a rule is applicable, change the inference rule item of that line to this compound rule, and change the reason pointers to the lines as far back in the logical predecessor relation as possible. For example, if A is used as reason in the justification that one particular inference is applicable, and we have two previous proof lines:

p1 ⊢ A∧B
p2 ⊢ A

where p1 precedes p2 in the proof, then the reason pointer should point to p1.

After the completion of the test above, we go through the whole proof again and delete the lines not used in the new proof (i.e. they are not reachable from the concluding line following the new reason relation backward). The assumption set of the remaining lines must be adapted correspondingly too. Notice that now definitions (axioms) and lemmas, since they do not appear

anymore as assumption proof lines, disappear from the assumption set of proof lines too. They are no longer taken as part of the premises, instead, they become part of the calculus. More elegant algorithms could be developed to narrow the candidates of possibly applicable compound rules. But the brute force method is itself not that inefficient because the number of axioms and theorems which a proof line depends on is always of a reasonable size.

3. Ordering Proof Trees

Having raised input Gentzen proofs to proofs at the conceptual level, we now proceed to translate them into so called message sequences. A message sequence is in fact a sequence of message units in internal code, each of which corresponds roughly to a sentence in the output paragraph. This sequence will later be taken over by the tactical component and translated into natural language. A linearization of the proof lines is therefore necessary. But instead of a direct linearization as e.g. in [Chester 76], we first organize the proof into an ordered tree. While the parents-children relation represents the basic inference relation (therefore the root of the tree is the conclusion of the theorem), the order on the children is based more on other structural and pragmatic relations. Notice as a tree, there might be much redundancy of nodes (proof lines). Obviously, an ordered tree of this kind not only provides us with a total order (through a simple post-order traversal), but also preserves all information of the original proof structure, for example the subproof structures. In the next section we will show how decisions on "what to say" for each single proof step are made, using among other things, this structural information.

Before going into concrete definitions and algorithms, let us first make clear what kind of constraints there are on the order of presenting a proof. First of all, there is the *logical constraint* that says reasons must be first proved before they can be used to justify any inference steps. As a matter of fact, this is nearly the sole constraint used in previous proof transformation systems[Chester 76]. However, there are another two kinds of important constraints which we found extremely useful and want to introduce in this section.

The first one is what we call *structural constraints* . This is based on the observation that many structural Gentzen inference rules demanding more than one reason impose an inherited order on their reasons, i.e. reasons are usually always derived in a particular order, although there is nothing logical forbidding doing it another way round. For example, in a proof applying the inference rule "CHOICE", we nearly always first derive

$\exists xP(x)$

before we assume

$P(a) \vdash P(a)$

and then prove

$P(a),A \vdash Q$

and then finally, discharge the constant "a" and derive $A \vdash Q$ by the rule "CHOICE". Logically it makes no difference if we prove $P(a),A \vdash Q$ before deriving $\exists xP(x)$. The same holds for the rule "CASE". We usually first derive

$A \vdash F \vee G$

before proceeding to check the two cases:

$F \vdash H$ and

$G \vdash H$

and finally derive by the rule "CASE"

$A \vdash H$.

This order is, of cause, closely related to the "forward chaining" proof development style that we take in our system. And forward chaining style is in fact the plainest and the most commonly used style in mathematical text books.

A closer investigation of proof examples reveals that a mere combination of logical and structural constrains normally does not suffice to impose a total ordering. Therefore another rather pragmatic constraint, the *focus mechanism*, is also used in our system. The focus mechanism was first developed in natural language understanding systems. It proved to be an effective computational tool in discourse interpretation. The global focus manipulation mechanism and the immediate focus manipulation mechanism were discussed in [Grosz 77] and [Sidner 79], respectively. McKeown [McKeown 85] first adapted the immediate focus mechanism of Sidner and incorporated it in her text generation system Text. When more than one choices concerning information inclusion or ordering still remain after the application of all other text structuring rules, the focus mechanism will be tried to single out a choice. Similarly, the focus mechanism in our system will be employed in such a secondary way as well.

In general, global focus refers to the fact that we usually center our attention on a particular object (collection) throughout a consecutive set of utterances. Immediate focus, on the other hand, describes the way in which our attention shifts or remains constant over two consecutive sentences. While we may use the immediate focus in the introsentential treatment which is one of the planned work, the global focus is already integrated with the other two constraints in our system to try to set up a total order for proof trees. It is in fact the first time that the global focus mechanism is used in a text planner in such a concrete manner.

The general observation is that, the focus phenomena discovered in natural language discourses also occur in a rather similar way in a proof presentation. As far as complying with the logical and structural constraints, consecutive proof lines always tend to center around one particular object (collection), in other words, properties of the same object are normally grouped together. That is to say, once we have started talking about a collection of particular objects, useful and derivable properties of this collection will be derived concecutively before the proof turns to properties of other objects. The argument will be made more precise by the following definitions.

We first briefly define the concept of a proof tree. Informally a proof tree is a tree with proof lines (or conjunctive subexpressions of proof lines) as nodes. The logical inference relation is then represented by the parents-children relation. The root is the conclusion of the whole proof. Leaves are assumptions, including premises of the theorem and assumptions introduced during the development of a proof which are discharged later. Assumptions and subproofs (subtrees) are possibly duplicated in order to build a tree instead of a graph. We decide not to label duplications differently in the tree, instead we will talk about different occurrences of nodes in the sequel.

Definition of Global Focus

Let N be an arbitrary node in a proof tree. The subtree rooted at N, which is a subproof, will be denoted by N too. The global focus of a subproof N is defined as follows:

$focus(N)=F \cup E$

where

$F=\{x \mid x$ is an object mentioned in subproof N$\}$

$E=\{x \mid \exists y\ y \in F$ and x is "associated" to y$\}$.

Notice the concept of focus is introduced with respect to structural objects. More precisely, Frame-like entity-constituent relations are considered. For our purpose, we use such a strategy that if an object is mentioned and hence in focus, all associated objects (for example, all its parts) are also in focus. At present we do not distinguish between explicit and implicit focus [Grosz 77]. While it might be improper under other circumstances, it works well for our mathematical application. Relations between objects currently considered include set-elements relation, algebraic structure-carrier set and -operator relation. A Frame-like internal representation will be used for such relations. A partial order $\subseteq_{focus}$ is also defined on focus spaces:

Let S1, S2 be two arbitrary focus spaces, the following holds

$S1 \subseteq_{focus} S2 \Leftrightarrow \forall x\ x \in S1 \Rightarrow x \in S2 \vee \exists y\ (y \in S2 \wedge x \subseteq y)$

This means elements of S1 are either elements of S2, or , in the case of set elements, a subset of some elements of S2. More elaboration is needed if objects with more complex structures are taken into account.

Now we can restate the previous discussion as follows: if we have started a proof, and at a certain point we have several alternatives to chose from as the next step, we should chose the one which remains in the same focus space, as far as this is allowed by the other two constraints. Here we have two points to make: in the first place the focus mechanism only plays a secondary role, the basic proof structure is determined by the logical and structural constraints. In the second place, concerning its practical applicability, we must first have a "start" point, i.e. a "start" focus space within which to continue, in order to apply the focus constraint. This "starting" focus space in turn must be set up by the other two constraints, as we will show. In our system, these three constraints are integrated in the following way in order to provide a total order.

First of all we build an initial proof tree from a proof already raised to the conceptual level. With the guarantee that the logical constraint has been satisfied, we start from the root node and proceed in a pre-order manner to try to enforce an order on children of every node. At each node we will first try those structural constraint rules. For example if we arrive at a node N with a proof line

$A \vdash Q$

which is derived by the rule "CHOICE" from the following two proof lines

$P(a),A \vdash Q$

$\exists x\ P(x)$

attached to node N_1, N_2 respectively, an order will be enforced on the two children N_1 and N_2 such that N_1 follows N_2. Beside this enforcement of order, some particular nodes of some subproofs are "marked", indicating that these proof lines should appear at the very beginning of the corresponding subproof (No conflictions have been encountered so far, therefore no conflictions handling mechanism is considered at the moment). In our case a node in subproof N_1 attached with a proof line $P(a) \vdash P(a)$ is marked, indicating that after the derivation of the existence of objects with certain properties, we usually first introduce such a constant, and then proceed to prove some other properties, and then discharge this constant again. Notice because we are working downward from the root, this marking that is often far "beneath" the current node provides frequently an "initial" focus space within which to continue.

Concretely these markings will be used at nodes where no existing structural rules are applicable. Suppose we arrive at such a node N with children N_1, N_2,..., N_k. And moreover suppose that we now have within the subtree N a node n marked by the previous process. This means that N must be first said and therefore it establishes an "initial" focus space. Now if possible, we are going to first proceed in this focus space before turning to another. A most straightforward thought might lead us to try to single out a child N_j satisfying

$$focus(n)=focus(N_j)$$

and make it the first child. The actual algorithm is neverthless a little more complicated in two ways: firstly, we do not always demand equality of the focus spaces, i.e. we claim that the proof will tend to continue in a subproof with the smallest focus space satisfying

$$focus(n) \subseteq_{focus} focus(N_j).$$

Secondly, if the focus mechanism can not directly link a marked node to a child N_j of the current node N, it generally suffices if it will link to a node which occurs exclusively in N_j. N_j will be picked out as the first child of N since the focus mechanism says the proof should at least complete a part of it, and since we usually do not interrupt subproofs (it is indeed guaranteed by the post-oder traverse manner in the production of a linear message sequence from an ordered tree later.).

With the above discussion on the general underlying principles in mind we give the following algorithm which enforces an order on the initial proof tree. Let rule(N) denote the inference rule used in deriving the proof line attached to node N, formula(N) the proof line, and order(N) the number of children of node N, respectively. While working through the proof tree, we have in fact as intermediate data structure a partially ordered tree, if we combine the post-order of the tree with the order imposed on children of nodes. In the sequel $N_1 < N_2$ denotes that node N_1 precedes node N_2 in a (partially) ordered proof tree. The concept of **duplication** in an ordered tree which is used in our algorithm is defined as follows:

Node n is a duplication node if there is a node n' attached with the same proof line such that $n' < n$. Notice it follows that all nodes in a subtree rooted by a duplication node are necessarily duplication nodes, therefore we can talk about duplication subtrees. Intuitively, a node n is a duplication node if its proof line has already been proved in the preceding context. Therefore, for the convenience of discussion, nodes attached with proof lines introduced by the "HYP" rule (including premises of the theorem) are also considered as duplication nodes.

Ordering Algorithm

Start at the root of the initial proof tree. Let N denote the current node under processing. If N is not a leaf node, execute rule 1, 2, 3 consecutively.

1. The *structural* rules

1.a. If rule(N)="CHOICE", formula(N)=A $\vdash$Q, order(N)=2. Let N_1, N_2 be children of N such that

formula(N_1)=P(a), A $\vdash$ Q and

formula(N_2)=$\exists$x P(x)

then make N_2 precede N_1, denoted as $N_2 < N_1$. Furthermore, mark node n with proof line P(a)$\vdash$ p(a) in subtree N_1.

1.b. If rule(N)="CASE", formula(N)=A $\vdash$ H, order(N)=3. Let N_1,N_2, N_3 be children of N such that

formula(N_1)=A $\vdash$ F $\vee$ G

formula(N_2)=A, F $\vdash$ H

formula(N_3)=A, G $\vdash$ H

then make N_1 precede both N_2 and N_3, denoted as:

$N_1<N_2$ and $N_1<N_3$

Notice no order between N_2 and N_3 is specified by this rule. Furthermore mark node n with proof line F $\vdash$ F in N_2 and n´ with proof line G $\vdash$ G in N_3, respectively.

1.c. If rule(N)="DED", formula(N)=A $\vdash$ F$\Rightarrow$G, order(N)=1. Let N_1 be the child of N_1 such that

formula(N1)=A, F $\vdash$G

then mark node n with proof line F $\vdash$F in N_1. Notice no order is specified by this rule. This marking information might later cooperate with the focus mechanism. The same holds for rule 1.d.

1.d. If rule(N)="IP", formula(N)=A $\vdash$F, order(N)=1. Let N_1 be the child of N such that

formula(N_1)=A , $\neg$F $\vdash$ $\perp$

then mark node n with proof line $\neg$F $\vdash$ $\neg$F in N_1.

2. The *focus constraint* rule

If order(N) $\geq$ m > 1 and N_1, N_2, ..., N_m are remaining unordered, non-duplication children of N. Let n be a marked non-duplication node occurs in at least one of the subtrees N_j $1 \leq j \leq m$. A particular child N_i will be picked out to precede the other children if we can find a node n´ such that

1´. n´ is a logical successor of n, i.e. n occurs in a subtree rooted by n´

2′. n′ occurs exclusively in the subtree Ni

3′. for any other logical successor n′′, focus(n′)$\subseteq_{focus}$focus(n′′)

That N_i is picked out means:

$N_i < N_j$ $\quad 1 \leq j \leq m$ and $j \neq i$.

Repeat rule 2 on the remaining child list as long as possible, until it is empty or there is no marked node usable.

3. IF order(N) $\geq m > 1$ and $N_1, N_2, ..., N_m$ are remaining unordered, non-duplication children of N. Rearrange them in an order

$N_1' < N_2' < ... < N'_m$

such that

$size(N_1') \geq size(N_2') \geq ... \geq size(N'_m)$

This last pragmatic rule says "longer" subproofs usually precede shorter ones so that conclusions of subproofs will not be too far away from the point where they are used as reasons. The size function is at the moment defined as:

size(N)=number of non-duplication nodes in the subtree N

+ number of largest duplication subtrees in N

where largest duplication subtrees are duplication subtrees without larger duplication subtrees containing them. They are counted as one proof line in the size function since they are all proved in the "preceding context" and will be simply used.

4. Repeat the whole process recursively on every child of the current node N.

Let us now return to our example and illustrate how the algorithm works. The initial proof tree constructed from the raised proof (proof 2 in appendix) and the resulting ordered tree are given in Fig. 2 and Fig. 3, respectively. Duplication nodes are shadowed in both trees. In the initial tree, only "premise nodes" are shadowed because no order is yet defined. In the resulting ordered tree however, all the duplication nodes are shadowed. Furthermore, only the root nodes of duplication subtrees are explicitly shown in the resulting tree. Numbers inside circles indicate the corresponding proof line or the corresponding conjunctive subexpression of a proof line. Thus for example, 1.4 indicates the 4th. conjunctive subexpression (from left to right) of the first proof line. In Fig. 3, children of nodes are drawn according to the imposed order from left to right as well.

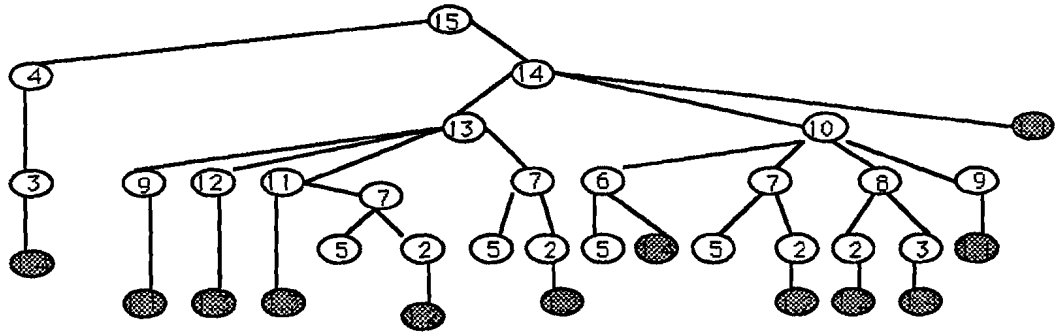

Fig. 2 Initial Tree

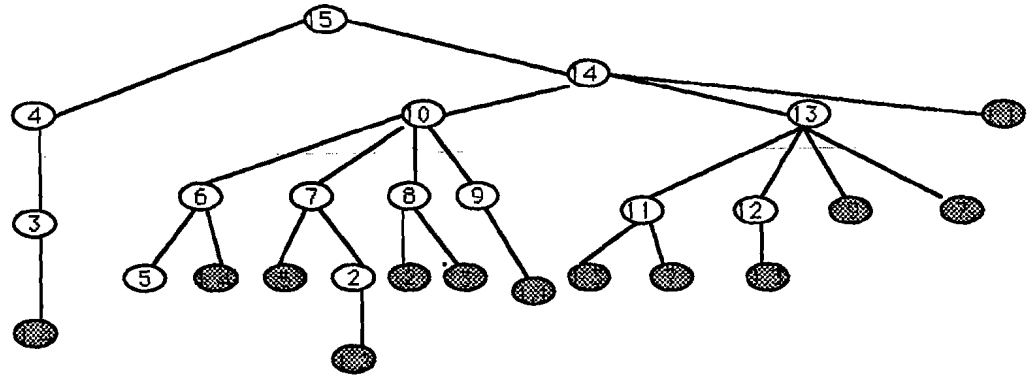

Fig. 3 Ordered Tree

The ordering process starts from the root node 15 in Fig. 2. Rule 1.a is applicable. Thus an order is imposed such that node 4 < node 14. And node 5 in the subtree rooted by node 14 is marked. Notice there are several duplications of node 5, which introduces a new constant u such that $u \in U$. The subtree rooted by node 4 has no further branchings and is hence already ordered. Now the algorithm arrives at node 14. A domain specific inference rule is used in the derivation of line 14 and no structural ordering rule is applicable. We have to try the focus constraint rule. Calculating the focus spaces of the marked node 5 and its logical successors node 6 and node 7 we have:

focus(node 5)=$\{U\} \cup \{x \mid x \in U\}$

focus(node 6)=focus(node 5)

focus(node 7)=focus(node 5) $\cup \{F\} \cup \{x \mid x \in F\}$

Note for set objects their elements are the only "associated" objects. Apparently node 6 has the smallest focus space. Checking the tree we find that node 6 has a unique occurrence in the subtree rooted by node 10, so an order is imposed such that node 10 < node 13. A similar argument makes node 6 the first child of node 10. The other three children of node 10 are ordered simply by rule 3 by decreasing size. Notice the ordering and the shadowing of duplications must be done hand in hand. After the shadowing the subtree rooted by node 13 can be ordered very simply too by rule 3.

The whole focus mechanism is still at the experimental stage. Much investigation on the appropriate definition of the focus space is underway. In addition, we are also considering incorporating a local focus mechanism similar to the one described in [McKeown 85]. Some of the

ordering decisions made by rule 3 based on subproof size may be justified by the local focus as well.

4. Proof Unit Model and Reference Choices

Once we have imposed an order on the proof tree, a simple post-order traversal will be used to produce a linearized message sequence which will be taken over by the tactical component and realized in natural language. During the traverse, decisions about "what to say" must be made at each node, or in other words, for each proof step. Generally, for each proof step a message unit of the format

«inference-rule, reasons, proof-line» 4.1

will be generated which contains the three necessary components of each inference step: the inference rule justifying this particular inference step, the already proved proof lines used by the inference rule as reasons, and the newly derived proof line itself. While the proof line will usually be handed over unchanged to the tactical component (one exception will be mentioned at the end of this section), there are alternative *reference choices* both for the inference rule and the reasons. The three reference choices we have identified and are dealing with in our system are:

1. The *explicit* form: this is the case where we may decide to indicate explicitly which inference rule we are using, which might be translated as "by the definition of unit element", "by the uniqueness of solution" etc., for corresponding domain-specific rules. For structural Gentzen rules we have such appropriate translations as well.

2. The *omit* form: in these cases simply a word such as "thus", "therefore" will be used.

3. The *implicit* form: this is between the two extremes of explicit and omit. By an implicit form we mean that although nothing is said directly as to the inference rule, an implicit hint to the inference rule is nevertheless given in the translation of the reasons (or of the proof line itself). For example, if we have a translation of a reason that reads "since 1 is the unit element of F", we might decide to omit the translation of the inference rule which would read "by the definition of unit". Another possible hint form is using a hint word. For example "similarly" is used to indicate that the same rule is used as in the previous proof step.

Similarly, three reference forms are found for reasons:

1. The *explicit* form: for example, reasons proved "far before" will usually be explicitly repeated.

2. The *omit* form: reasons can be omitted if they have just been proved or mentioned in other proof steps.

3. The *implicit* form: similar to inference rules, reasons can be "hinted" implicitly. For example, consider the case where we infer 1u=u from the two reasons unit(1 F OP) and $u \in F$ using an inference rule derived from the definition of unit. Suppose unit(1 F OP) is proved somewhere in the previous context. Now directly after proving $u \in F$ we may say: "by the definition of unit, 1u=u". Here the reason unit(1 F OP) is implicitly refered to by an explict translation of the inference rule

(compare the implicit reference choice for inference rules described above, which is exactly in the opposite direction).

Let us first consider the reference choices for inference rules. Unlike the reasons, the explicitness or implicitness of referring to an inference rule at a particular node is irrelevant to the position of the node in the whole proof. That means it is less concerned with the proof context than with the user's familiarity with the particular inference rule, or in the case of domain-specific inference rules, with the corresponding definition or theorem. With no sophisticated user model at our disposal at the moment, the following relatively simple strategy is used for the reference of inference rules:

Reference Choice Rules for Inference Rule

1. Reference Choices for Gentzen Inference Rules

We assume that our user is familiar with the standard Gentzen Calculus and therefore translate all Gentzen inference rules in a predescribed way. This means that each Gentzen rule will either be given always implicitly or always explicitly, unchanged throughout the whole proof and in all proofs. No additional assumption on user's knowledge is taken into account. The standard reference rules are:

1.a. All *structural* Gentzen rule will be explicitly given.

1.b. All *non-structural* Gentzen rules will be omitted (a word like "thus", "hence", etc. will be used).

Notice that sometimes a reference choice for an inference rule can have influence on the choice of reference for the corresponding reasons. For example, an explicit translation of the case rule will be something like: "From case A and case B, it follows....", which requires that the two reasons, represented here by their names, should be at least implicitly mentioned. The reader will find a definition of various ways of mentioning a reason afterwards when we discuss decisions on reason references.

2. Reference Choices for Domain-specific Inference Rules

In general domain-specific inference rules the user is familiar with can be omitted. Otherwise they will be explicitly indicated. Users are assumed to be familiar with definitions and theorems of the "underlying theory" which our current theory is based upon. For example, when we are reasoning about properties of group theory we assume that the users are familiar with basic set theory and omit the inference rules derived from the definitions or theorems of basic set theory. The information about theory levels and their interdependency will be stored in our knowledge base. A current theory level pointer must be set as well. Thus we have:

2.a. All domain-specific inference rules derived from a definition or theorem of a *lower level* will be omitted.

2.b. For each inference rule derived from a definition or theorem of the *current level*, try first to find an implicit form. If not possible, an explicit indication will be given. Notice no distinction is made between different rules derived from the same theorem or definition.

Therefore, what finally appears at the position "inference rule" in the message format 4.1 will be one of the following:

1. A name of one of the structural Gentzen rules,

2. A name of a definition or theorem of the current theory level, or

3. "omit", currently for all other cases, much refinement of this rule is needed.

A natural language interpretation will be given in the dictionary for each of this. This, however, will be the task of the tactical component and will not be discussed in this paper.

Now let us turn to choices on reason reference forms. The three reason reference forms listed above may have reminded us of the various ways to refer to objects in natural language discourses: using a pronoun, by name, using a full describtion and so on. Similar to their counterpart in natural language discourse, as has been mentioned, decisions on reason reference forms are also normally context sensitive. On the one hand the concrete physical distance plays an important role, for example a reason proved "far before" will very likely be repeated. Choices of reference forms for reasons, on the other hand, depend on another at least equally important factor as well. That is the proof structure. In general every proof is composed of some subproofs, which in turn may consist of their own subproofs. This structure, though simpler, has some striking similarities with the discourse context structure discovered by R. Reichman for natural language discourses [Reichman 85]. While discourse context theory was used to decide whether a particular referent is currently in foreground or background of the focus of attention, in order to choose an appropriate reference form, we are going to discuss how the proof structure will affect a choice of reference forms for a particular reason. For example, a human mathematician reading a proof is normally supposed to still remember the assumptions of the particular subproof he is working on, although the last mention of that assumption may be physically quite "far away". In other words, the assumptions are in foreground and an explicit repetition should be avoided. In the rest of this section we will first give a definition of our proof structure, espetcally the definition of its basic units, which we call **proof unit**. Then an algorithm will be given to show how reference forms will be affected by both this proof structure and the physical distance.

In general every proof structure is a recursive structure of basic components which we call proof units. In our ordered proof tree a proof unit is simply a subtree. But apparently not all subtrees can be appropriately taken to be a proof unit. Proof units are intuitively subtrees which a human mathematician will accept as a subproof, which is in fact a conceptually integral unit inside a proof on which he will once concentrate during the construction or reading of a proof. First of all we claim that the structural inference rules of Gentzen natural deduction calculus provide us with an excellent means of dividing proofs into units. For example, the CASE rule naturally divides a proof into three units: the whole proof itself and the two cases. Secondly we claim that a unit is also completed when an "important" intermediate result is reached, which is often the result of applying an inference rule derived from the theory of current level. On the contrary, applications of inference rules derived from lower level are usually considered trivial and do not form a conceptual subproof. The following is a formal definition of proof units:

Definition of Proof Unit

1. Every subtree in a proof tree rooted at a node derived by an "important" inference rule, an inference rule derived from a definition or theorem of the current level is a proof unit.

2. Every subtree in a proof tree rooted at a node derived by the Gentzen inference rule **DED, IP** is a proof unit; in both cases, mark the "assumption nodes" (see introduction) such that they will not be included in any further subordinate unit. Marking assumptions means that they are assumptions of the whole proof unit and should not be included in any subordinate units.

3. Every subtree in a proof tree rooted at a node derived by the Gentzen inference rule **CASE** is a proof unit, all its cases are proof units. Mark "assumption nodes" in all cases, respectively.

4. Every subtree in a proof tree rooted by a node derived by the Gentzen inference rule **CHOICE** is a proof unit, its two reason subtrees are proof units. Mark the assumption node in the subtree rooted at a node with proof line in form of P(a),A $\vdash$ Q.

Notice the marking here has nothing to do with the previous marking of duplication nodes. In graphs this new marking is then shown by darking the number inside the circles (see node 5 in Fig. 4).

As it is shown in Fig. 4, five proof units are identified by applying the above rules on the ordered proof tree in Fig. 3. Unit p1 is the whole proof and thus the outermost proof unit. p2 and p3 are identified by the CHOICE rule and are directly subordinate to p1. Notice the "assumption node" in p3, node 5, is marked and is therefore not included in the subordinate unit p4. p4 and p5 are justified by the fact that node 10 and node 13, their roots, are both derived by an "important" domain-specific rule, which comes from the definition of solution of equation in the theory of current level.

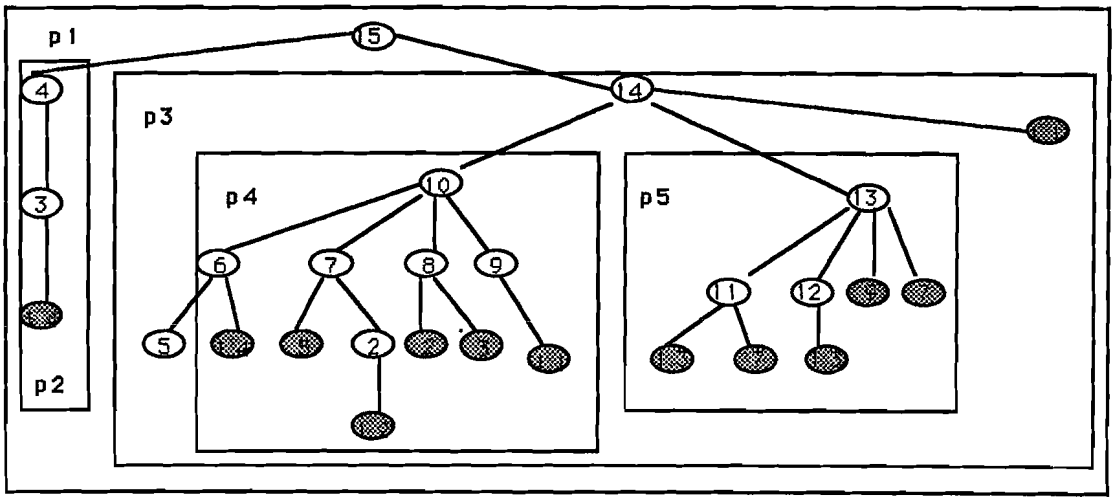

Fig. 4 Proof Units

The ultimate objective of developing a proof unit model is to study the relations between proof units and to reveal their impact on the choices on reason reference forms. The approch we take here is similar to the discourse context theory of Reichman [Reichman 85]. While he identified seven types of context spaces, only four proof unit types are currently defined in our proof unit model.

Proof Unit Types

Let us suppose that we are traversing an ordered tree and generating one message unit at every node. With respect to the current node in an ordered proof tree under processing we define:

1. The *active proof unit* is the smallest proof unit containing the current node. There is exactly one active proof unit at a time.

2. The *controlling proof unit* is the smallest proof unit containing the active unit. There is exactly one controlling proof unit at a time, except when the active proof unit is an outmost proof unit.

3. *Precontrol proof units* are proof units containing the controlling proof unit.

4. *Closed proof units* are proof units lying before the active proof unit in the ordered proof tree, i.e. nodes inside any closed proof unit have already been processed when the traverse procedure reaches the current node.

Now take node 11 in Fig 4 as the current node. Then p5 and p3 are the active unit and the controlling unit, respectively. p2 and p4 are already closed and p1 is the only precontrol unit at the moment. Similar to the fact that the reference choices in natural language discourses depend on the status of the context space in which the corresponding referent is last mentioned, our proof unit model claims that the status of the proof unit in which a reason is proved or last mentioned has a strong influence on the reference form of the reason. More precisely, the status of the smallest proof unit in which the reason last appears determines, together with the "physical distance" between this last appearence and the current point, the reference form of the reason. In the sequel, as far as not otherwise indicated, the proof unit of a reason is always assumed to be this smallest unit.

For example, if a reason is last mentioned or proved in the active proof unit, which is the proof unit a human proof reader/writer is currently working on, it is supposed that this reason should still remain in his focus of attention. If the reason is in a closed unit, on the other hand, it is very likely that the reason has already been moved out of the reader/writer's focus of attention. It is claimed that this focused attention space has a strong influence on the reference forms. Reasons that still remain in the focus of attention when the process reaches the current point will be considered "*structurally close* " to the current point, otherwise they will be considered "*structurally far* ". In the following we give rules assigning this "structural closeness":

Assignment of Contextual Status for Reasons

1. Reasons in the *active* proof unit are structurally close.

2. Reasons in the *controlling* proof unit but not inside any closed unit are structurally close.

3. Reasons that are root nodes of immediate subordinate closed proof units are structurally close. Other reasons in closed proof units are structurally far.

4. Reasons in precontrol proof unit are far (We are still experimenting with this rule.).

Rule 3 means that only the conclusions of closed subproofs still remain in the focus of attention. In addition, as a special treatment, premisses of theorem (which are shadowed in our ordered tree, see Fig. 3) will be defined as both structurally far and far in distance. Now the following rules provide a criterion of reason reference choice.

Reference Choice Rules for Reason

1. If a reason is structurally close and near in distance, it will be omitted.

2. If a reason is structurally close but far in distance, first try to find an implicit form, if not possible, use an explicit form.

3. If a reason is structurally far but near in distance, first try to find an implicit form, if not possible, omit it.

4. An explicit form will be used if a reason is both structurally far and far in distance.

Here physical distance means the textual distance between the last mentioning of a reason and the current sentence where the reason is used. This distance is currently calculated in an ad-hoc way. Notice that the result of applying rule 2 and rule 3 depends on the fact that an implicit form is possible, which often interacts with the choice of reference for the inference rule.

Now it is easy to combine the *Reference Choice Rules for Reason* and the *Reference Choice Rules for Inference Rules* to construct an algorithm to produce a message sequence from the ordered tree: we first declare all nodes associated with a proof line which is the premise of the theorem as duplicate, they are also initialized as both structurally far and far in distance. Then we simply traverse the whole ordered tree in a pre-order manner and apply the two sets of rules at each non-duplication node. Notice at each node only one rule in both rule sets will be appliable. Of course there is nevertheless still the possibility of non-determinency if a rule allows more than one reference form, for example, the second and third rule for reason reference. Very often a decision made by one rule set will help to select between remaining alternatives in the other. For example, suppose the process arrives at node 3, the first node in fact. Since node 1.4, its unique reason, is a premise of the theorem and thus both structurally far and far in distance, an explicit form is chosen. This choice on reason references leads to the choice of an implicit form for the inference rule, which is one of the alternatives in rule 2.b. This is because this reference rule is derived from the definition of unit, which is a definition of the current level, and our knowledge base tells us that mentioning unit(1_u U OP) in the reason reference provides an implicit hint that some rule about unit is used. This produces the first piece of message in our message sequence:

«omit, unit(1_u U OP), $1_u \in$ U»

When this is not the case, that means either there are more than one alternatives in the appliable rules of both set, or the decision in one rule set does not help to narrow the alternatives in the other set, the process is allowed to choose arbitarily at present. For example suppose we are now at node 6. The reason node 1.4 is now structurally far but near in distance. It can be either realized in an implicit form or be omitted. And there are two reference possibilities for the domain-specific inference rule as well. An artbitrary choice will produce:

«def.unit, omit, $u1_u$ =u»

where an implicit form and omit form are used for the infererence rule and the reason, respectively.

Finally we want to indicate that although normally the proof lines will be handed over into the message units unchanged, there is one exception. It is omitted in the translation of a proof step justified by the Getzen structural rule "CHOICE", since the proof line is essentially not changed,

only an assumption is discharged. The translation might read "the result is independent of the choice of constant a.". The whole message sequence of the ordered tree in Fig. 4 is given as follows:

Message Sequence Produced

1. «omit, unit(1_u U OP), $1_u \in U$»
2. «omit, omit, $\exists x\ x \in U$»
3. «suppose, omit, $u \in U$»
4. «def.unit, omit, $u1_u = u$»
5. «omit, subgroup(U F OP), $U \subseteq F$ »
6. «omit, omit, $u \in F$»
7. «**similar**, omit, $1_u \in F$»
8. «omit, group(F op),semigroup(F op)» this step will be omitted in the future
9. «omit, omit, solution(u u 1_u F OP)»
10. «omit, unit(F 1 OP) $\wedge$ $u \in F$, u1=u»
11. «omit, omit, $1 \in F$» (10 and 11 should be merged into «omit, unit(F 1 OP) $\wedge$ $u \in F$, u1=u $\wedge$ $1 \in F$», because premise of 9 subsumes premise of 10)
12. «omit, omit, solution(u u 1 F OP)»
13. «uniqueness-of-solution, omit, 1= 1u»
14. «independent-of(u), omit, omit»

Just like the focus mechanism in the last section, much refinement is needed for both the definition of proof unit and for the reference choice algorithms. For example it may be necessary that we distinguish "directly" before from near in distance. In addition, more elaboration is needed for the reference forms themselves. For example we are just considering splitting the "omit" form for reasons further into two, since the surface translation of the cases where the reasons are just proved and where the reasons are just mentioned in another proof step is usually different. In the first case it may be something like "it follows," or simply "thus" or "therefore", in the second case, however, the word "and" might be used to indicate the same reason is used. In other words, better categorization of reference forms may be necessary. Finally, we are also considering mechanisms that merge message units for consecutive proof steps when appropriate. For example, we are experimenting with merging consecutive message units with identical "reason" or "inference rule".

5. Conclusion and Future Work

In this section, we first give a possible natural language translation and compare it with the original text, then we will mention briefly the future developments we are planning.

The theorem (translated from German originally given in [Deussen 71]):

Let F be a group and U∈F a subgroup, if $1_u \in U$ is a unit element of U, then $1=1_u$.

Proof (A Possible Natural Language Translation):

(1) Because 1_u is the unit element of U, $1_u \in U$. (2) Thus $\exists x\ x \in U$. (3) Now suppose u is an arbitrary element of U. (4) By the definition of unit, $u1_u=u$. (5) Because U is a subgroup of F, $U \subseteq F$. (6) Thus $u \in F$. (7) Similarly, $1_u \in F$. (8) In addition, F is a semigroup because it is a group. (9) Now we have proved that 1_u is a solution of the equation ux=u in F. (10) On the other hand, because 1 is the unit element of F and $u \in F$, u1=u and $1 \in F$. (11) Thus 1 is also a solution of the equation ux=u in F. (12) By the uniqueness of solution in a group, $1=1_u$. (13) This conclusion is independent of the choice of the element u.

Sentences are numbered for convenience of discussion. Notice besides message unit 10 and 11, which are combined into one sentence (sentence 10), each message unit is translated into a separate sentence.

Proof: (a translation from the original German text given in [Deussen 71])

1_u is the solution of the equation $u \cdot x=u$ in U. Since $x \in U \subseteq F$ and F is a group, equation ux=u has a solution in F, namely x=1. By the uniqueness of solution, $1=1_u$.

The original proof in German [Deussen 71]:

1_u ist die Lösung der Gleichung ux=u in U. Da $x \in U \subseteq F$ und F Gruppe ist, hat ux=u eine Lösung in F, nämlich x=1. Die Eindeutigkeit der Lösung bringt $1=1_u$.

The distinction between the original proof and the proof generated by our system is still very clear. Our proof resembles a proof where a mathematician decides not to leave out a single proof step. This means that the small proof steps appearing in the input Gentzen proof and left out in our output text in fact never appear in a proof written by a human mathematician. If we examine the original text of [Deussen 71] carefully, however, we find that the proof is in fact somewhat above this *conceptual* level. Some conceptual proof steps are omitted and must be reconstructed by the reader himself, if he wants to be strict. Only the steps most crucial to the whole proof (and some steps related to the crucial steps) are explicitly given (in our case the application of the unique solution theorem, and the premises required by the theorem.). Because these crucial steps correspond closely to the crucial steps in Alan Bundy's proof plan [Bundy 87][Bundy et al 88], we may say that proofs found in mathematical text books are sometimes more or less on the *plan level.*

As has been pointed out in the corresponding sections, more elaboration is necessary for all the three main components of our proof transformation system. In this section, we are going to touch some of the future developments that are not part of any of the three components.

1. Introsentential treament: the formula of proof lines is usually extremly simple in mathemetical applications. Therefore we at present hand over the proof lines unchanged to the tactical component. But essentially treatment of the proof line itself is also necessary if the translation is to be really

natural. Because one proof line is normally translated into one sentence in natural language, we call it introsentential treament. We believe no general solution to this problem is possible at present, since the treatment of even one of the most simple formula schemes

$$P1 \wedge P2 \wedge ... \wedge Pn$$

equals the problem of generation from a knowledge base. The information units picked out from a knownledge base are in fact connected together in a conjunctive manner. In general for each special field of mathematics an "assertive" or "descriptive" scheme similar to those in [McKweon 85] is required. More precisely, typical schemes for presenting mathematical facts must be identified and specified. For example the "from whole to parts" strategy will enforce a translation "F is a group with unit element 1" instead of a translation "1 is the unit element of F and F is a group". The local focus mechanism should be used to ensure coherence as well.

2. The Overlapping of natural language interpretation and "natural" coding.

A dictionary containing the natural language interpretation of all predicates used will be employed by the tactical component. Depending on the particular coding we have, overlapping of natural language interpretation for different predicates may occur. This, if translated without simplification, leads to redundancy in the final natural language output. The reason underlying this phenomenon is that the predicates used in most logic based knowledge bases are "larger" than word senses. So in general a simplification is needed. At the same time we are testing the "naturalness" of the coding too. The following is our new coding procedure:

Conceptualization: A precise definition of the problem field is first defined, including definitions of objects (object types), which is currently a Frame-like structure[Kerber 89][Brachman et al 1983] and definitions of all relations (tuples they should contain).

Axiomatisation: Axioms and definitions are formulated in predicate logic such that the conceptualization is a model of it. Although an axiomatisation such that the conceptualization is its unique model is not always available, model A will be considered "better" as model B, if A is "smaller" than B. Natural codings will usually reduce duplications of natural language interpretation.

Finally we want to briefly indicate that the whole process described in this paper is not restricted to first-order predicate logic. It can be used nearly unchanged for natural deductions written in higher order logic. Indeed, the problems are very often first coded in a higher order formal language and then translated down to first order, before the MKRP theorem prover can prove it. But afterwards it must be again transformed back to higher order before the transformation described in this paper can take place.

Acknowledgement

I would like to thank Christoph Lingenfelder and Manfred Kerber for many discussions. I would like also to thank Norbert Reithinger who introduced me into the field of Text Generation. I would

like to thank especially Jörg Siekmann and Norbert Eissinger for the many discussions and constructive suggestions on earlier versions of this paper.

6. References

[Bundy 87] A. Bundy(1987), The Use of Explicit Plans to Guide Inductive Proofs. Dai Research Paper No. 349, Dept. of AI, Univ. of Edinburgh.

[Bundy et al 88] A. Bundy, F.van Harmelen, J. Hesketh, A. Smaill (1988), Experiments with Proof Plans for Induction. Dai Research Paper No. 413.

[Brachman et al 1983] Ronald J. Brachman, Richard E. Fikes, and Hector J.Levesque, KRYPTON: A Functional Approach to Knowledge Representation. In: R.J. Brachman and H.J. Levesque ed: Readings in Knowledge Representation.

[Chester 76] D. Chester, The Translation of Formal Proofs into English. AI 7, 1976.

[Deussen 71] P. Deussen, Halbgruppen und Automaten. Springer-Verlag, 1971.

[Fatemen et al 88] Richard Fatemen, Alan Bundy, Richard O'keefe, Leon Sterling (1988) Commentary on: Solving Symbolic Equations with Press. Dai Research Paper No. 357.

[Gentzen 35] G. Gentzen, Untersuchungen über das logische Schließen I, Math. Zeitschrift 39, 1935.

[Grosz 77] B.J. Grosz, The Representation and Use of Focus in a System for Understanding Dialogs. In B. J. Grosz et al ed: Readings in Natural Language Processing, 1986.

[Hayes 1979] Patrick J. Hayes, The Logic of Frames. In R.J. Brachman and H.J. Levesque ed: Readings in Knowledge Representation 1985.

[Kerber 89] M. Kerber, A Frame Based approach to Representing Mathematical Concept. SEKI-Report, to appear.

[Lingenfelder 86] C. Lingenfelder, Transformation of Refutation Graphs into Natural Deduction Proofs. SEKI-Reprot, SR-86-10, 1986

[Lingenfelder 88] C. Lingenfelder, Structuring Computer Generated Proofs. SEKI-Report SR-88-19..

[MKRP 84] Karl Mark G Raph, The Markgraf Karl Refutation Procedure, Memo-SEKI-Mk-84-01, Universität Kaiserslautern, 1984.

[McDonald 83] D.D.McDonald, Natural Language Generation as a Computational Problem. In : Brady/Berwick: Computational Models of Discourse, MIT Press, Cambridge, MA, 1983.

[McKeown 85] K. R. McKeown, Text Generation. Cambridge Univ. Press, 1985.

[Reichman 85] R. Reichman, Getting Computer to Talk Like You and Me. Discourse Context, Focus, and Semantics (An ATN model). MIT Press, 1985.

[Sidner 79] C.L. Sidner, Focusing in the Comprehension of Definite Anaphora. In B. J. Grosz et al ed: Readings in Natural Language Processing, 1986.