Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-91-13

International Workshop on Terminological Logics

Schloß Dagstuhl, May 6-8, 1991

Organizers:

Bernhard Nebel **Christof Peltason** **Kai von Luck**

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

International Workshop on Terminological Logics

Organizers: Bernhard Nebel, Christof Peltason, Kai von Luck

DFKI-D-91-13

International Workshop on Terminological Logics

Schloß Dagstuhl, Germany
May 6 – 8, 1991

Organizers:

Bernhard Nebel	Christof Peltason	Kai von Luck
DFKI	TU Berlin	IBM Germany

Contents

Preface

The second international workshop on Terminological Logics was held at Schloß Dagstuhl[1], near Saarbrücken, Germany, on May 6-8, 1991. The workshop was the follow-up event to the "Workshop on Term Subsumption Languages" held in New Hampshire, in October 1989 (cf. AI Magazine 11(2), 1990).

Terminological Logics consists of a family of representation formalisms that have grown out of the KL-ONE knowledge representation system. Unlike some other areas of knowledge representation, in this field the aspects of theoretical work (semantical foundations, complexity), system-oriented work (implementations), and application-oriented work are all dealt with within one community, as documented by the variety of talks at this workshop.

The workshop itself brought together 40 invited participants currently working in the field, and served to provide a snapshot of the current state of research, showing that there has been a lot of progress in the last several years. The theoretical area has advanced to a point where only a few questions concerning the core formalism remain open. The current trend seems to be to integrate more functionality and other formalisms.

The material compiled here consists of most of the contributions of the participants, listed in alphabetical order of the submitting authors. In character with the informal nature of the "work"shop, these papers sketch personal interests, work in progress, or summaries of research results rather than being fully elaborated articles. The complete set of presentations at the workshop can be seen from the program given on page 2.

In addition to the scheduled sessions, there were a number of informal meetings for exchanging ideas and planning future collaborative work, including one about future system standards and standard notation (see also the proposal on page 120). This should make the exchange of ideas, systems, and knowledge bases, and the maintainance of a test corpus easier in the future.

The program was rounded off by an overview talk by Ron Brachman on the past and future development of Terminological Logics (the issue of finding a good name for the field is still in discussion), and a panel debate on aspects of the relationship between "Theory and Practice". In order to promote communication between people working in the field a mailing list (tlc@isi.edu) was established.

We would like to thank the Dagstuhl foundation for inviting us, our affiliated organizations for their support, and finally all participants for their active engagement in the workshop.

July, 1991

Bernhard Nebel (DFKI)   Christof Peltason (TU Berlin)   Kai von Luck (IBM Germany)

[1]Schloß Dagstuhl is the site of the recently founded International Conference and Research Center for Computer Science, sponsored by the German Society for Computer Science (GI), and three universities (Saarbrücken, Kaiserslautern, and Karlsruhe), and financed by the the federal states of Saarland and Rheinland-Pfalz.

Workshop Program

Sunday, May 5

afternoon	Arrival

Monday, May 6

8:45 - 9:00	Introduction Bernhard Nebel (DFKI, Saarbrücken)
9:00 - 10:45	**Algorithms and Complexity** Franz Baader (DFKI, Kaiserslautern) Francesco Donini (Univ. Roma) Werner Nutt (DFKI, Kaiserslautern) Peter F. Patel-Schneider (AT&T Bell Labs) Klaus Schild (TU Berlin)
10:45 - 11:15	*Break*
11:15 - 12:30	**Principles of Modeling** Howard W. Beck (Univ. Florida) Alfred Kobsa (Univ. Saarbrücken) Joachim Quantz (TU Berlin) Sonia Bergamaschi (Univ. Bologna)
12:30 - 14:00	*Lunch*
14:00 - 15:45	**Implementation Techniques** Carsten Kindermann (TU Berlin) Eric K. Mays (IBM, New York) Bob MacGregor (ISI, Marina del Rey) Deborah L. McGuinness (AT&T Bell Labs)
15:45 - 16:15	*Break*
16:15 - 17:30	**Probability and Defaults** Jochen Heinsohn (DFKI, Saarbrücken) Lin Padgham (Linköping Univ.)
18:00	**Keynote Address** Ron Brachman (AT&T Bell Labs)
19:00	*Buffet*

Tuesday, May 7

8:30 – 10:00	**NL-Applications** Jürgen Allgayer (Univ. Saarland) Amadeo Capelli (Univ. Pisa) Manfred Gehrke (Siemens, München)
10:00 – 10:30	*Break*
10:30 – 12:00	**Unification-based Systems** Guiseppe Attardi (Univ. Pisa) Roland Seiffert (IBM, Stuttgart) Gert Smolka (DFKI, Saarbrücken)
12:00 – 12:30	Organisational Issues
12:30 – 14:00	*Lunch*
19:00	*Dinner*
20:00	**Discussion: Theory and Practice** Bob MacGregor (ISI, Marina del Rey) Hector Levesque (University of Toronto)

Wednesday, May 8

8:30 – 9:45	**Hybrid Extensions** Bernhard Pfahringer (Univ. Wien) Enrico Franconi (IRST, Povo) Albrecht Schmiedel (German Heart Center, Berlin)
9:45 – 10:15	*Break*
10:15 – 12:00	**Technical Applications** Rüdiger Klein (AdW, Berlin) Luca Spampinato (Quinary, Milano) Bill Swartout (ISI, Marina del Rey)
12:00 – 12:30	Organisational Issues
12:30 – 14:00	*Lunch*
14:00 – 15:30	**Workshop Summary and Outlook**
18:00	*Dinner*

Experiences in 'Hybridification': Enhancement of a Term Subsumption Language to Cover Plural and Quantified Terms.

J. Allgayer
University of Saarbrücken
FB 14: Dept. of Computer Science
6600 Saarbrücken 11, Germany

1 Introduction

In the context of natural language processing, term subsumption languages (TSL) are widely used for several tasks, from lexicon structuring to domain modeling. One portion of the problems in NL dialog systems is the necessity to represent *and* reason about dialog contributions that construct step by step the context of the dialog, e.g. the intruduction of entities the participants are talking about. This spans from descriptions of entities, over relations they have been proposed with, to attitudes and modes they were proposed in. Not only needs all this be represented in a way expressible and powerfull enough to cover all representational means as well as processing demands from all parts of the overall system, but also an adequate reasoning about these representational terms is necessary.

The facet of the described problem field which will be described in more detail in this paper deals with the requirement of a language (called SB-TWO) that should a) utilize a TSL (SB-ONE, [Kobsa89], [Profitlich90]), but b) extend/enhence it with representational means to deal with descriptions of plural terms and relationships between them, and a flexible way of quantification.

Considering the state of the art in NL processing (see, for example, Discourse Representation Theory ([Kamp81]), File Change Semantics ([Heim82]), Situation Semantics ([Barwise/Perry83]) or the Generalized Quantifier Theory ([Barwise/Cooper81])), there is a clear tendency in NLP either to base the processing directly on a formalism taken from linguistics, or to define a well-founded underlying theory according to linguistic criteria, in order to get a system which behaves in a well-defined manner, and which is extendable for dealing with new phenomena.

Therefore, in most cases the NLP formalisms are based on first order predicate logic; but this leads to a mismatch between the properties of the natural language to be characterized and those of the knowledge representation language that should describe them.

Notions like quantifier, variable, sense and reference, intension and extension, ... are all technical (...) notions introduced by philosophers and logicians. They are not part of the data of natural language. It just might be *that some or all of them cut across the grain of the phenomena in unnatural ways, generating artificial problems and constraining the space of possible solutions to the genuine puzzles that language presents.* ([Barwise/Perry83, p.xi f.])

On the other hand, notions like these are very useful – and, as we believe, are indespensable – with respect to an internal formal knowledge representation language. Therefore, we look for an application of the above-mentioned notions that fits "the grain of the NL phenomena" in the best possible way.

And this, in fact, is the intention of the Generalized Quantifier Theory (GQT, [Barwise/Cooper81]): to provide a notion of (formal) quantifiers that describes quantifiers as they occur in natural language.

2 Natural Language Determiners

Research concerning General Quantifiers has been (and still is) mainly motivated by three arguments:

A1 Not all Natural Language Determiners are expressible by formulas of first order predicate logic (FOL); propositional determiners like "more than half of" would need higher order expressions;

A2 in order to formally describe the relations between a predicate (verb) and its arguments (noun phrases) in a compositional semantics, a formalism is needed that treats all possible structures of NPs in a unique manner;

A3 different determiners behave in different ways with respect to the deductions that can legally be drawn from the proposition the determiner is involved with. One would wish to formalize this observation in order to restrict the whole set of relations denoted by Natural Language Determiners (called DET from now on), as well as to subdivide DET into subclasses for which specific deductions hold.

The GQT's view of determiners is to see them as relations between two sets: the set of individuals denoted by the NP's noun (which is referred to as the Basic predicate), and the set of individuals denoted by the VP (the Central predicate). The main result shows that any determiner in ((Det Basic) Central) that satisfies the fundamental priciples of conservativity, extensionality, variety, and quantity (see below) can be defined via the cardinalityt of two sets: $|$ Basic $\cap$ Central $|$ and $|$ Basic $\setminus$ Central $|$.

We see that the definition of meaning of determiners is totally independent from the underlying model, which, in turn, is an argument to lower the treatment of determiners into the knowledge representation language.

Especially results of GQT with respect to (A3) are most interesting for NLP systems, as we shall see after the next section.

On the representational basis, we need to be able to express sets and relationships between sets in order to lower the properties and characteristics of various NL determiners into the representation formalism. The notion of *whitness set* from GQT can be seen as the representative of the set of sets a determiner is mapped onto. Thus, being able to handle sets appropriately enables us to treat determiner processing " à la GQT". The representational prerequisites will be briefly discussed in section 4.

3 Formal aspects of semantic constraints of NL determiner

As GQT has shown, all NL determiners satisfy the following fundamental principles which therefore may be seen as linguistic universals.

> **Conservativity (CONS):**
> If Det in ((Det Basic) Central) describes a valid relation between the sets Basic and Central, then ((Det Basic)(Basic $\cap$ Central)) is valid as well.

This is important for the proper treatment of determiners in an NLP system (based on any KL-ONE family member and the open world assumption), because one never has any information about the extension of the Central predicate. But, as CONS tells us, the semantics of DET doesn't depent on the cardinality | Basic \ Central |.

The second principle fulfilled by all natural language determiners guarantees context-neutrality:

> **Extension (EXT):**
> If Det in ((Det Basic) Central) describes a valid relation between the sets Basic and Central in a model A and there is a Model A' with A $\subseteq$ A', then ((Det Basic) Central) is valid in A' as well.

What this means is that the extension of the discourse – which is done permanently in a system with underlying open world asumption – does not change the semantics (the truth value) of the quantified proposition, as long as it doesn't concern the denotation of the predicates involved.

The principle of Quantity is concerned with the specific interpretation function underlying the model and is often called the principle of topic-neutrality:

> **Quantity (QUANT):**
> If ((Det Basic) Central) describes a relation between the sets Basic and Central in a model A, and there is a function π representing a bijection from A to A', then ((Det π(Basic)) π(Central)) is an equivalent relation in A'.

Thus, the interpretation of a determiner-specific relation is independent of the specific properties of the elements of discourse. This again supports the view that the handling of determiners should be a task of the knowldege representation formalism itself.

> **Variety (VAR):**
> If Basic is non-empty, then there exist two predicates Central and Central' such that ((Det Basic) Central) and $\neg$((Det Basic) Central') hold.

This means that no element of DET is trivial in the sense that the determiner relation either holds for any pair of elements of the universe or for none.

Beyond these principles which are fulfilled by all elements of DET, there are some characteristics of specific subclasses of DET that define *inferential properties* of the structures corresponding to the determiners on the level of world knowledge representation.

To give an idea, take these as an example:

Transitivity: ((Det Basic) Central) $\cap$ ((Det Central) Central') $\Rightarrow$ ((Det Basic) Central')

Example: "every", counterexample: "two"

Symmetry: ((Det Basic) Central) $\Rightarrow$ ((Det Central) Basic)

Example: "some", counterexample: "every"

For NLP systems with knowledge representation formalisms based on conceptual hierarchies, the most interesting properties of determiners are those that influence its inferential behaviour with respect to the underlying terminological basis.

> **Monotonicity:**
> If a determiner Det is *upward/downward monotone* then if ((Det Basic) Central) describes a valid relation between the sets Basic and Central, and there is a Superconcept/Subconcept Central' that subsumes Central, then ((Det Basic) Central') is valid as well.

Upward monotone determiners are, for example,"some" and "at least ten", where "at most ten" is not.

> **Persistency:**
> If a determiner Det is *upward/downward persistent* then
> if ((Det Basic) Central) describes a valid relation between the sets Basic and Central, and there is a Superconcept/Subconcept Basic' that subsumes Basic, then ((Det Basic') Central) is valid as well.

To summarize, the requirements necessary to cope with NL determiner processing in an adequate fashion appear to be the following:

- R1 a laguage that is able to express determiner properties and to compute the hierarchy of determiner classes;
- R2 a knowledge representation language that is able to express assertions in which NL quantification (i.e., the usage of determiners) is required;
- R3 integration of and access to knowledge bases (KBs) that describe the properties of determiners (i.e., the DET classification KB);
- R4 integration of inference capacities arising from the usage of DET with the reasoning procedure of the overall inference mechanism (i.e., intelligent use of the KB which holds the determiner dependent inference rules);
- R5 the ability to express sets and their relationships *as well as* set elements next to each other.

4 The representational foundations

In order to satisfy requirement R5, an extension of the existing knowledge representation formalism (and system) SB-ONE ([Kobsa89, Profitlich90]) has been defined. This formalism (called **SB-ONE$^+$**) regards the universe of discourse as consisting of entities and all possible groupings of those entities, i.e., $U = D \cup P(D)$, where D is the set of domain elements, and $P(D)$ denotes the powerset. Thus, an instatiation of a concept, called an instance, may be of type *set* or *element*, respectively. And, quite naturally, the formalism allows to express relationships among these entities of our "world", e.g. *subset, superset,* or *element-of* relationships, and maintains the relationships stated implicitly (and computable via the transitivity of such relations). For a more detailed explanation of **SB-ONE$^+$**, see [Allgayer90].

5 A determiner processing system

The system's layout shown in Figure 1 gives some insight into the approach taken for the solution of our problems and requirements. The PROLOG meta-interpreter MOTHOLOG[1] combines the KL-ONE knowledge representation paradigm with the representational and computational power of logic. It does this by allowing for the integrated use of knowledge expressed in an inheritance network formalism (in our case, **SB-ONE$^+$**), as well as the description and use of inferential knowledge expressed in inference rules.

MOTHOLOG allows for access to network based knowldege bases that simulates the solution generating process underlying the PROLOG reasoning mechanism. Queries to such KBs are translated into continuation-based reentry procedures that compute a set of solutions and deliver on demand one solution after the other. Thus, a net-query can be seen as a simulated sequence of queries to an ordinary PROLOG database which includes backtracking after failure or unsatisfying proof results. This allows for the use of network based knowledge bases within a logic-oriented language.

Equipped with these feasabilities, the DET classification KB can be used to determine which determiner was used and what kind of properties are declared for it.

Knowing this, we can use the appropriate inference rules for this type of determiner. Again, MOTHOLOG inference rules are capable to visit the appropriate KBs when interpreting the determiner dependent inference rules (DDIs). DDIs themselve express the semantic properties of determiners as described in section 2. For example, a DDI expressing the upward persistency of a certain determiner (which is a member of the class of persistent determiners!) looks like

```
(gen-quant (?Det ?Var ?Basic) (?Central)) :-
        (detclass ?Det persisten/upward);
        (supsumes ?SubBasic ?Basic);
        (gen-quant (?Det ?Var ?SubBasic) (?Central))
```

It expresses the fact that by using an upward persistent determiner in an expression, the Central predicate can be replaced by a more general one, preserving the truth conditions for the newly generated expression.

[1]MOTHOLOG is an acronym for 'MOre THan Ordinary proLOG'

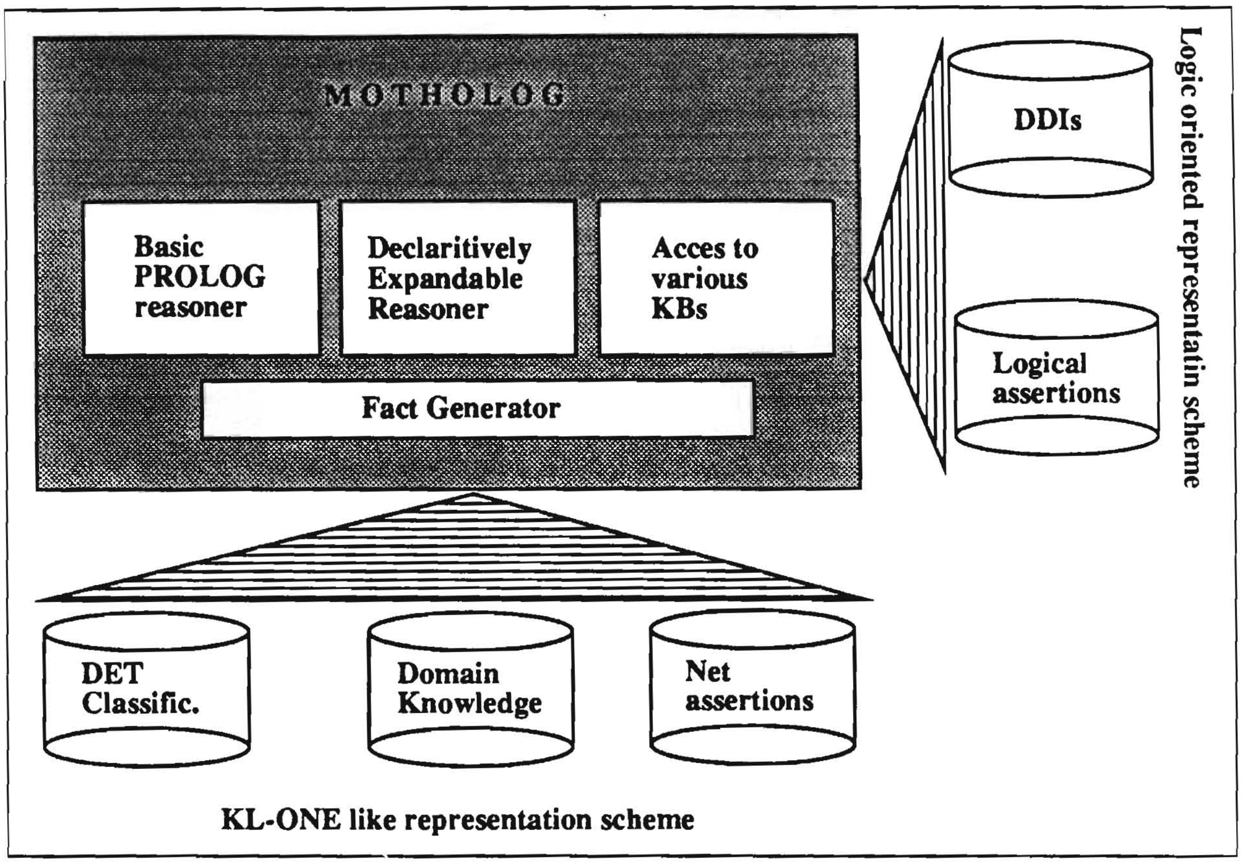

Figure 1: Architecture of MOTHOLOG

The predicate **detclass** implements the connection to the knowledge represented in the determiner classification knowledge base, and **gen-quant** couples the usage of determiners and their adequate processing.

References

[Allgayer/Reddig90b] J. Allgayer and C. Reddig. What KL-ONE Lookalikes Need to Cope with Natural Language – Scope and Aspect of Plural Noun Phrases. In K.H. Bläsius, U. Hedstück and C.-R. Rollinger (eds.), *Sorts and Types in Arificial Intelligence*, Berlin, Heidelberg, New York, London, Paris, Tokyo, Hong Kong: Springer, 1990.

[Allgayer90] J. Allgayer, **SB-ONE**$^{+}$ – dealing with sets efficiently. In: Proc. ECAI 90.

[Barwise/Cooper81] J. Barwise and R. Cooper. Generalized Quantifiers and Natural Language. Linguistics and Philosophy, 4:159–219, 1981.

[Barwise/Perry83] J. Barwise and J. Perry. Situations and Attitudes. Bradford Books. Cambridge, MA: MIT Press, 1983.

[Heim82] I. Heim. The Semantics of Definite and Indefinite Noun Phrases. PhD thesis, Univ. of Massachusetts, 1982.

[Kamp81] H. Kamp. A Theory of Truth and Semantic Representation. In: J. A. G. Groenendijk, T. M. V. Janssen, and M. B. J. Stokhof (eds.): Formal Methods in the Study of Language, pp. 277–322, Amsterdam: Mathematical Centre, 1981.

[Kobsa89] A. Kobsa. The SB-ONE Knowledge Representation Workbench. Memo 31, Univ. Saarbrücken, 1989.

[Profitlich90] SB-ONE: Ein Wissensrepräsentationssystem basierend auf KL-ONE. Master's thesis, Univ. of Saarland, 1990.

Cyclic, Transitive, and Concrete Extensions of Concept Languages

Franz Baader
German Research Center for AI (DFKI)
Postfach 2080
W-6750 Kaiserslautern, Germany
e-mail: baader@dfki.uni-kl.de
phone: (+49 631)205-3457

Abstract

The purpose of this note is not mainly to describe particular results on extensions of concept languages, but rather to illustrate the evolution of research which lead to these results, and to show how they are connected.

1 Terminological Cycles in $\mathcal{FL}_0$

Cyclic definitions are often prohibited in terminological knowledge representation languages because, from a theoretical point of view, their semantics is not clear and, from a practical point of view, existing inference algorithms may go astray in the presence of cycles. In [Baa90c] terminological cycles are considered in a very small KL-ONE-based language which allows only conjunction of concepts and value-restrictions. For this language, which will be called $\mathcal{FL}_0$ in the following, the effect of the three types of semantics introduced by [Neb87, Neb89, Neb90]—namely, least fixed-point semantics (lfp-semantics), greatest fixed-point semantics (gfp-semantics), and what he called descriptive semantics—can be completely described with the help of finite automata. These descriptions provide a rather intuitive understanding of terminologies with cyclic definitions, and give insight into the essential features of the respective semantics. In addition, one obtains algorithms and complexity results for subsumption determination. The results of [Baa90c, Baa90d] may help to decide what kind of semantics is most appropriate for cyclic definitions, not only for the small language $\mathcal{FL}_0$, but also for extended languages. As it stands, the greatest fixed-point semantics comes off best. The characterization of this semantics is easy and has an obvious intuitive interpretation. Furthermore, important constructs—such as value-restriction with respect to the transitive or reflexive-transitive closure of a role—can easily be expressed.

However, the results obtained in [Baa90c] have two major drawbacks which we intend to overcome in [Baa90a]. First, the language $\mathcal{FL}_0$ is too small to be sufficient for practical purposes. As shown in [Baa90d], the results can be extended to the language $\mathcal{FL}^-$ of [LB87], and it seems to be relatively easy to include number-restrictions. However, as soon as we also consider disjunction of concepts and exists-in-restrictions, the unpleasant features which lfp-semantics had for $\mathcal{FL}_0$ (see [Baa90c, Baa90d]) also occur for gfp-semantics in this larger language. If we should like to have general negation of concepts, least or greatest fixed-points may not even exist, thus rendering fixed-point semantics impossible. Second, the characterization of gfp-semantics for $\mathcal{FL}_0$–though relatively easy and intuitive—still involves notions from formal language theory such as regular languages and finite automata.

2 Union, Composition, and Transitive Closure of Roles

In [Baa90a] it is shown that the concept defining facilities of $\mathcal{FL}_0$ with cyclic definitions and gfp-semantics can also be obtained in a different way. One may prohibit cycles and instead allow role definitions involving union, composition, and transitive closure of roles. The regular languages which occur in the characterization of gfp-semantics for $\mathcal{FL}_0$ can directly be translated into role definitions in this new language. This proposes a way of retaining, in an extended language, the pleasant features of gfp-semantics for $\mathcal{FL}_0$ with cyclic definitions without running into the troubles caused by cycles in larger languages.

Starting with the language $\mathcal{ALC}$ of [SSS]—which allows one to use negation, conjunction and disjunction of concepts as well as value-restrictions and exists-in-restrictions—cyclic concept definitions are disallowed in [Baa90a], but instead the possibility of role definitions involving union, composition, and transitive closure of roles is added. In contrast to other terminological KR-systems which incorporate the transitive closure operator for roles, a sound and complete algorithm for concept subsumption is given in [Baa90a]. The connection between role definitions involving union, composition, and transitive closure of roles on the one hand, and regular languages over the alphabet of all role names on the other hand is also important for this algorithm.

3 Terminological Cycles and Concept Equations in $\mathcal{ALC}$

Since $\mathcal{ALC}$ contains general negation of concepts, descriptive semantics is the only meaningful semantics for cyclic definitions in this language. It is easy to see that the transitive extension of $\mathcal{ALC}$ mentioned in the previous section is not equivalent to $\mathcal{ALC}$ with cyclic definitions interpreted with descriptive semantics. Nevertheless, the algorithm developed for subsumption testing in the transitive extension can be used to decide subsumption for $\mathcal{ALC}$ with cyclic definitions.

More general, it is even possible to decide subsumption with respect to general concept equations by using this algorithm (see Section 6 of [Baa90a]). A general concept equation is an axiom of the form $C = D$ where both C and D may be complex concept descriptions. In contrast to these general equations, the usual T-Box axioms always have simple concept names on the left hand side, with the additional restriction that any name may occur only once as a left hand side. Please note that the implication rules used in many terminological KR-systems (e.g., in BACK or CLASSIC) can easily be expressed by such general concept equations. In fact, an implication rule $C \Rightarrow D$ is logically equivalent to the concept equation $C \sqcap D = C$ where $\sqcap$ stands for conjunction of concepts.

4 Integrating Concrete Domains

The extension described in this section was motivated by an application in a mechanical engineering domain (see [BBK+91]).

A drawback which concept languages based on KL-ONE have is that all the terminological knowledge has to be defined on an abstract logical level. On that level one can e.g. describe the concept Woman as "humans who are female", and represent it by the expression Human $\sqcap$ Female. In many applications, however, one would like to be able to refer to concrete domains and predicates on these domains when defining concepts. Examples for such concrete domains are the nonnegative integers, the real numbers, or also non-arithmetic domains, and predicates could be equality, inequality, or more complex predicates. In the above example, one might think that being human and female is not enough to make a woman. As an additional property one could require that she should be old enough, e.g., at least 21. Thus one would like to introduce a new role **age**, and define Woman by an expression of the form Human $\sqcap$ Female $\sqcap$ $\geq_{21}$(**age**). Here $\geq_{21}$ stands for the unary predicate $\{n; n \geq 21\}$ of all nonnegative integers greater or equal 21. Stating such properties directly with reference to a given concrete domain seems to be easier and more natural than encoding them somehow into abstract concept expressions.[1] Though this drawback already appears in natural language processing, it becomes even more important if one has other applications in mind. For example, in a technical application the adequate representation of geometrical concepts requires to relate points in a coordinate system. For that purpose one would e.g. like to have access to real arithmetic.

In [BH90a] we propose a scheme for integrating such concrete domains into concept languages rather than describing a particular extension by some specific concrete domain. We define a terminological and an assertional language, and consider the important inference problems such as subsumption, instantiation, and consistency. The formal semantics as well as the reasoning algorithms are given on the scheme level. The algorithms generate subtasks which have to be solved by a special purpose reasoner of the concrete domain. A concrete domain for which these subtasks are solvable is called admissible in [BH90a]. In contrast to existing KL-ONE based system, the algorithms will be not only sound but also complete, provided that the concrete domain in question is admissible.

5 Combining the Extensions

For many applications (an in particular also for the above mentioned application in mechanical engineering) it is desirable to have both access to an admissible concrete domain and transitive closure of roles. We have mentioned above that adding one of these two facilities to a concept language such as $\mathcal{ALC}$ leaves the interesting inference problems decidable. However, the situation changes if we want to have both facilities in one language.

[1]See e.g. [BS85], Section 9.2, where so-called Structural Descriptions are used to encode the concrete predicate "less than one hour". From a computational point of view, Structural Descriptions are as bad as Role Value Maps which cause undecidability of subsumption [SS89].

If, starting with $\mathcal{ALC}$, we allow only transitive closure of functional roles (without union or composition of roles) and integrate the admissible concrete domain $\mathcal{R}$ (which stands for real arithmetic) then the subsumption problem becomes undecidable.

This can be shown by reducing the Post Correspondence Problem to the subsumption problem for this language. The reduction uses only very simple predicates from real arithmetic, namely equalities between linear polynomials in at most two variables (see [BH90a] for details).

References

[Baa90a] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. Research Report RR-90-13, DFKI / Kaiserslautern, 1990. A short version will appear in the Proceedings of the IJCAI'91.

[Baa90b] F. Baader. A formal definition for the expressive power of knowledge representation languages. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 53–58. ECAI, 1990.

[Baa90c] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, volume 2, pages 621–626. AAAI, 1990.

[Baa90d] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. Research Report RR-90-01, DFKI / Kaiserslautern, 1990.

[BH90a] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. Research Report RR-91-10, DFKI / Kaiserslautern, 1990. A short version will appear in the Proceedings of the IJCAI'91.

[BH90b] F. Baader and B. Hollunder. Kris: Knowledge representation and inference system. Technical Memo TM-90-03, DFKI / Kaiserslautern, 1990. To appear in the SIGART Bulletin.

[BBK+91] A. Bernardi, H. Boley, C. Klauck, P. Hanschke, K. Hinkelmann, R. Legleitner, O. Kühn, M. Meyer, M.M. Richter, F. Schmalhofer, G. Schmidt, and W. Sommer. ARC-TEC: Acquisition, representation and compilation of technical knowledge. In *AVIGNON 91*, 1991.

[BS85] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[LB87] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[Neb87] B. Nebel. On terminological cycles. KIT Report 58, KIT Group, Technische Universität Berlin, 1987.

[Neb89] B. Nebel. Terminological cycles: Semantics and computational properties. In *Proceedings of the Workshop on Formal Aspects of Semantic Networks*, 1989. Two Harbors, Cal.

[Neb90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 1990.

[SS89] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. J. Brachman, editor, *First International Conference On Principles of Knowledge Representation and Reasoning*, pages 421–431, 1989.

[SSS] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. To appear in Journal of Artificial Intelligence, 47, 1991.

A Conceptual Clustering Alogrithm for Semantic Data Models *

Howard W. Beck

University of Florida
12 Rogers Hall
Gainesville, FL 32611
INTERNET: hwb@beach.cis.ufl.edu

The Intelligent Information Retrieval Project at the University of Florida has been exploring the application of term subsumption languages to database management and natural language processing. The CANDIDE semantic data model was developed as an adaptation of KANDOR, and explored the use of classification as a query processing technique. An information retrieval system was developed using CANDIDE which featured a natural language query interface.

Recently research has focused on expanding term subsumption languages to more accurately reflect fundamentals of category formation. A general conceptual clustering algorithm has been developed which augments deductive reasoning using subsumption with inductive reasoning by generating classes over sets of instances. The clustering algorithm is applied to several database applications including schema design, schema evolution, schema integration, view generation, and query processing. The database schema also supports represention of lexical knowledge by organizing large numbers of cases of word use. Lexical acquisition from cases is being explored.

1 Conceptual Clustering Algorithm

Conceptual clustering techniques based on current theories of cateorization [6, 5] provide a way to design database schemas which more accurately represent classes. In this approach, classes are treated as complex clusters of concepts rather than simple predicates. An important service provided by the database is determining whether a particular instance is a member of a class. A conceptual clustering algorithm aids in building classes by grouping related instances and developing class descriptions. The resulting database schema addresses a number of properties of categories including default values and prototypes, analogical reasoning, exception handling, and family resemblance.

Class cohesion results from trying to resolve conflicts between building generalized class descriptions and accommodating members of the class which deviate from these descriptions. This is achieved by combining techniques from machine learning, specifically

*This research is supported in part by grants from the Florida High Technology and Industry Council

1. Introduce a New Class
   1.1. Use SUBSUME and Classification to determine the relationship between the new class and existing class descriptions.
   1.2. Use Realization to determine which existing instances satisfy the new class description.
2. Introduce a New Instance
   2.1. Use Realization to place the new instance into classes for which the instance satisfies class descriptions.
   2.2. Use INTERSECT to identify other related instances. This may generate new classes, but is also needed in the next step.
   2.3. Use the Exception Condition to see if the new instance may be an exception to an existing class description.
   2.4. Based on a decision to place an exceptional instance into a class, use EVOLVE to modify the class schema.

Figure 1: Main components of the conceptual clustering algorithm

explanation-based learning and case-based reasoning. A subsumption function is used to compare two class descriptions. A realization function is used to determine whether an instance meets an existing class description. A new function, INTERSECT, is introduced to compare the similarity of two instances. INTERSECT takes two instances and generates a new class which is the minimal description which is satisfied by both instances. INTERSECT is used in defining an exception condition. Exception handling results in schema modification (EVOLVE).

The main components of the clustering algorithm are outlined in Figure 1. The purpose of the clustering algorithm is to assign instances to classes. In the process, existing classes may be modified (schema evolution) and new classes formed. The process is incremental in that each new instance or class is being added to an existing database. The structure (schema) of the database must be altered to account for the new instance or class. The process is conceptual in that it is based on a comparison of the structures of database objects which represent concepts.

2 Database Applications

A conceptual clustering algorithm based on current theories of categorization should be used as the basis for organizing and maintaining databases classes. This results in databases classes which more accurately reflect real category structures. In this section, the usefulness of this approach in database applications is discussed. The topic is explored in greater detail in [2].

- Schema Generation and Evolution

  The techniques described in the previous section are most directly applicable to schema generation and maintenance. They can be used as tools to help database designers. Since the algorithm is incremental, new classes and instances can be added at any time. The database system evolves with each new addition in that the schema is modified as needed to accommodate the addition. The process is open-ended, leading to more complex, more accurate schemas as more information is added.

- Schema Integration

  The conceptual clustering algorithm can be used to generate a global schema which integrates several different databases. Anwar et al. [2] describes a procedure in which a schema is generated from the combined instances from several databases. Relationships among instance attributes between databases must be specified, such as synonomus relationships, set/subset relationships, and logical implication (such as the relationship between age and date of birth). Instances from several databases can then be clustered into a global schema based on specified goals and user preferences.

- Query Processing

  The conceptual clustering algorithm can be applied to query processing. Two such applications are processing queries with inexact answers [1] and providing intensional answers to queries.

  - Queries with Inexact Answers

  Analogical reasoning can be applied in situations where an inexact match to a query specification is desirable. Such may be the case when no instances match the query exactly. For example, it may be desired to find job candidates which most closely match the job description, though no candidate may match precisely. In case-based reasoning, it is a fundamental operation to retrieve cases which are somehow similar to the new case. Such queries will be more important as databases are used in analogical reasoning.

  In contrast to numeric or fuzzy sets approaches which ultimately rely on some distance metric and threshold to processing such queries, conceptual clustering retrieves instances which are structurally, semantically, and pragmatically similar to the query even though they may not match the query exactly. The query processor has both a deductive and inductive component. The deductive component finds exact matches in the traditional sense, and the inductive component identifies ways in which inexact matches may be considered similar. Ranking on similarity is done using the database taxonomy by which similar instances become members of the same class. Relative similarity is determined by depth in the taxonomy.

  Query processing is accomplished through conceptual clustering by representing the query as an object (either a class or instance), and using the clustering algorithm to determining the correct position of the query object within the taxonomy. The use of classification in exact query processing was discussed in an earlier paper [7] where classification was a purely deductive procedure. Queries are represented as new database classes. Subsumption relationships between the query class and other classes were computed to find the most specific classes to which the new query class belongs. Instances of these classes were then tested to see if they conformed to the restrictions stated in the query class. Those instances which satisfied the restrictions were retrieved as the result of the query.

Inexact query processing extends this procedure by considering partial matching between database instances and the query object. Partial matches can be determined through INTERSECT. The generated class tells how two instances are similar. Incrementally computing this relationship over the instances of a database results in a rich taxonomy of clusters in which related instances are grouped into the same classes. The result of a query is a new structural organization of the database schema which tells, 1⟩ Those instances which match exactly, 2⟩ Those instances which match inexactly, and 3⟩ A class taxonomy describing the relationships between the query and other instances. The taxonomy is also used to rank relative similarity between the query and retrieved instances.

- Intensional Answers to Queries

The conceptual clustering algorithm will build a taxonomy of subclasses subsumed by the new query class. These subclasses provide an intensional answer to the query which can be stated in terms of the data definiton language [2]. Thus, in addition to retrieving a set of instances related to the query, these subclasses would be available to summarize the categories to which these instances belong. This intensional answer would contain more information than simply displaying a list of instances which match the query, since relations among the instances are categorized by the subclasses.

A database schema design based on category theory more naturally represents the meaning of terminology, thus leading to an improved user interface since terms used to describe the data can be mapped onto database objects. In particular, we are using the data model discussed here to store a large lexicon for use in natural language query processing [3, 4]. The conceptual clustering algorithm is used to assist in building the lexicon by modifying representations for word meaning as new word usages are encountered.

- Automatic Generation of Views Based on Clustering Seeds

  Database views can be specified by giving a clustering seed. The clustering seed is a class description or small schema representing the desired concepts that guide the view creation process[7, 2]. This seed provides the clustering algorithm with a basis for generating a database schema which conforms to the desired view. Existing subclasses and instances related with the view are clustered beneath the seed. Since it is directly related to query processing, view creation can take advantages of the query capabilities just described.

3 Language Acquisition

The natural language processing component of the project is concentrating on lexical acquisition through case-based reasoning [3, 4]. The conceptual clustering algorithm supports retrieval of relevant cases. Database instances are used to represent cases. Language acquisition from similar cases is treated in the context of concept acuqistition and category theory. The main contributions from case-based reasoning include mapping new or unusual usage to related cases and determining default values over sets of cases for use in disambiguation. This work is being implemented as part of a information retrieval project involving language acquisition from corpus of text.

Expectation-driven parsing fails in interpretation of new or novel usage patterns since, by definition, new or novel usage patterns are precisely those which are not expected. An expression is ungrammatical if it fails to fit patterns described by a particular set of grammar rules. Yet language speakers are often able to glean understanding from such expressions and may readily acquire the new usage pattern. Case-based reasoning provides a way to overcome the brittleness of strictly rule-based natural language processing by providing an interpretation of unusual utterances. Case-based language understanding is a "language usage" theory. We understand an utterance because of its similarity to previous utterances, not because the utterance fits the mold of a general, idealized rule. Nevertheless, case-based language understanding should be treated as one part of a more general theory which balances similarity-based reasoning with rule-based reasoning.

Understanding an unusual utterance requires a global search of cases which may only partially match with the new utterance. Instead of (or more accurately, in addition to) comparing the utterance to general usage patterns, a case base of previous utterances provide a corpus of text which can be examined for relationships to the new utterance. The case base of text provides two techniques for analysis:

1. Finding cases is treated as a database query with complete or partial matching. Context (expectation) is use to specify the query initially.

2. Default values obtained by statistical measures over sets of cases is used to weight the retrieved cases. Usage patterns with the highest frequency are given priority.

Finding cases is admittedly a computationally complex task. This would not be such a problem where it done in parallel. Otherwise the use of indexing can speed performance as it has in other applications of case-based reasoning.

4 Summary

An information retrieval system of agricultural data, mostly text, is being constructed using the CANDIDE semantic data model [7, 8]. Natural language processing is being developed for querying the database, but eventually also to help in building the database by extracting information directly from text. Language acquisition is seen as a significant bottleneck in this process. Conceptual clustering is a fundamental component of both the database organization and language acquisition.

References

[1] T. Anwar and H. Beck. *Inexact Queries: A Conceptual Clustering Approach.* Technical Report, Database Systems Research and Development Center, University of Florida, Gainesville, FL, 1991.

[2] T. Anwar, S. Navathe, and H. Beck. *A Semantically Adaptive Modeling Interface for Schema Generation over Multiple Databases.* Technical Report TR-90-16, Database Systems Research and Development Center, University of Florida, Gainesville, FL, 1990.

[3] H. Beck. Language acquisition from cases. In R. Bareiss, editor, *Proc. DARPA Case-Based Reasoning Workshop*, Morgan Kaufmann, San Mateo, CA, 1991.

[4] H. Beck. A lexicon design based on theories of categorization. In U. Zernik, editor, *Proc. First International Lexical Acquisition Workshop*, AAAI Press/MIT Press, Cambridge, MA, 1989.

[5] H. Beck. *A Terminological Knowledge Representation System Based On Theories of Categorization.* PhD thesis, University of Florida, Gainesville, FL, 1990.

[6] H. Beck, T. Anwar, and S. Navathe. *A Conceptual Clustering Algorithm for Database Schema Design.* Technical Report TR-91-05, Computer and Information Sciences, University of Florida, 1991.

[7] H. Beck, S. Gala, and S. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *Proc. IEEE Fifth International Conference on Data Engineering*, Los Angeles, CA, 1989.

[8] H. Beck, P. Jones, D. Watson, and F. Zazueta. An expert database system for ornamental plants. *Agricultural Systems*, 31:111–126, 1989.

Subsumption for Database Schema Design

Sonia Bergamaschi and Claudio Sartori
CIOC-CNR
Università di Bologna

Abstract

The aim of this paper is to show the effectiveness of subsumption in the relevant field of database research of *schema* (terminology) *design*. After some general considerations, we briefly synthetize the results of two works dealing, respectively, with developing terminological logics for the so-called *semantic models* [3], and the more recent *complex object models* [2, 1], proposed in database environments. The developed idea is that, by extending database models with *defined concepts* and giving them a terminological logic formalization, it is possible to compute *subsumption*, thus allowing a formal definition of *consistency* and *minimality* of a schema to be given and an active tool supporting schema acquisition to be developed.

Primitive and Derived classes, Inheritance

The idea of subsumption and defined concepts might seem, at first glance, extraneous to database environment, where isa relationships between classes must be explicitly declared and a class description usually represents necessary conditions for the *extension* of a class, which is explicitly filled with individuals. As a matter of fact, we observe that defined concepts present similarities with *views* in RDBMSs (i.e. virtual relations, computed on the basis of base relations), with *derived subtypes* [5] of semantic data models, and with *derived types*, expressed by horn clauses, in deductive databases. The main difference is that such modelling primitives are mainly used at instance level as *derivation rules* to fill the corresponding relations (classes).

Furthermore, we observe that both primitive and defined concept semantics are useful because, usually, the upper levels of a conceptual schema are constituted by primitive concepts (no sufficient and necessary conditions are available), while lower levels are constituted by defined concepts. The limited expressivity of both tractable terminological formalisms and database models does not often allow the full definition of a class to be given: structural descriptions of KL-ONE have been excluded from concept descriptions for computational problems, and derivation rules as well as *integrity constraints* (complex necessary conditions) are used at instance level and are often hidden in a program. Nevertheless, there is a subset of DB derivation rules and integrity constraints which are already available or can be added to terminological formalisms without compromising tractability

Dipartimento di Elettronica, Informatica e Sistemistica - Viale Risorgimento 2 - 40136 Bologna, Italy - Tel: +39 51 644.3550-3548 - E-mail: {sonia,claudio}@deis64.CINECA.IT

(classes disjointedness, constraints on value domains, cardinalities, reference constraints), which make the introduction of *derived classes* (i.e. defined concepts) in databases quite convincing.

Another point is the different perspective on inheritance of the most recent complex object database models: at the schema level, we have types and primitive classes. Types denote data structures and extensions, i.e. domains of elements. Classes denote data structures and also extensions, i.e. collections of objects in a database. However, the extension denoted by a type is fixed and defined by the structure, while the extension of a class is user definable. Therefore the following equations hold respectively for classes, say C_1, C_2, their structures (*typ*) and types, say T_1, T_2:

$$\begin{aligned} C_1 \; isa \; C_2 &\Rightarrow I(C_1) \subseteq I(C_2) \\ C_1 \; isa \; C_2 &\Rightarrow typ(C_1) \preceq typ(C_2) \\ T_1 \preceq T_2 &\Leftrightarrow typ(T_1) \preceq typ(T_2) \Leftrightarrow I(T_1) \subseteq I(T_2) \end{aligned}$$

Notice that the $\preceq$ relationship, called *type refinement*, defined for types is syntactically computed in a way very similar to subsumption, and is easily extensible to derived classes.

By extending database models with derived classes, and giving them a terminological logic formalization, provided we are able to develop complete and tractable subsumption algorithms, we can develop an active tool for schema acquisition, preserving consistency and minimality of a schema. Consistency can be guaranteed for a taxonomy of only primitive classes: given a new class, by subsumption it is possible to compare its description with a given classes taxonomy and detect whether it is *incoherent* (subsumed by the empty class) and, if so, reject it. A more active role can be played if the taxonomy includes also derived classes: for a coherent class description a *minimal description* (i.e. a class re-written description on the basis of its *most specific generalization classes*) is computed, thus the class is placed in the correct position of a taxonomy. In this way, *equivalence* of classes is recognized, (i.e. different names and (or) syntactic descriptions which correspond to the same minimal description are detected) and *redundancies* with respect to a taxonomy can be avoided.

Subsumption for Semantic Models

In [3] it is shown that the data modelling primitives of the best known semantic models giving prominence to type constructors (Entity-Relationship, TAXIS, GALILEO) can be expressed with the terminological language $\mathcal{FL}^*$, whereas the data modelling primitives of the models giving prominence to attributes (FDM, DAPLEX, IFO) can be expressed with $\mathcal{FL}^*_{inv}$, which extends $\mathcal{FL}^*$ with inverse roles. $\mathcal{FL}^*$ is defined as follows:

$$\begin{aligned} C, D \longrightarrow \; & A \mid \neg A \mid C \sqcap D \mid \\ & \forall R.C \mid (\geq nR) \mid (\leq nR) \mid \\ & \forall A_t.V \mid (\geq nA_t) \mid (\leq nA_t) \end{aligned}$$

where V is defined as:

$$\begin{aligned} V \longrightarrow \; & integer \mid real \mid string \mid Vname \mid \\ & integer\text{-}range \mid real\text{-}range \mid Vname\text{-}range \mid \\ & Vname = (\langle atom_1 \rangle \ldots \langle atom_n \rangle) \end{aligned}$$

$\mathcal{FL}^*$ and $\mathcal{FL}^*_{inv}$ include attributes and value domains semantics, following database tradition. It is worth noting that in order to capture the semantic of the most expressive semantic models, we had to develop $\mathcal{FL}^*_{inv}$ ($\mathcal{FL}^*_{inv} = \mathcal{FL}^* + R^{-1}$) whose expressivity is equivalent to $\mathcal{PL}_1$, one of the two maximally polynomial languages recently defined [4] ($\mathcal{FL}^*_{inv} = \mathcal{PL}_1 + A_t$). Therefore, for both classes of semantic models, polynomial and complete subsumption algorithms can be developed and the goal of guaranteeing consistency and minimality of a schema can be achieved. The objective of schema minimality is obtained by the definition of *minimal description* of a class. Given a class description C and a terminology T, the minimal description of C, say C^{md}, is expressed as $C^{md} = C_1 \sqcap \ldots \sqcap C_i \sqcap C_d$, with $C_i \in MSG(C)$, which is the most specific generalizations set, and C^D is the *difference concept*, which can be univocally computed from C and $MSG(C)$, as is proved in [3].

Subsumption for Complex Object Data Models

Complex objects data models, recently developed in the DB area, are adopted in both Object Oriented Databases and Deductive Databases. The following description generalizes different models, trying to synthetize the most relevant modelling principles. At schema level, we have types and primitive classes, denoting complex structures and extensions. Complex structures are obtained by recursively applying any of the basic constructors to types: set ($\{\}$), tuple ($[]$), sequence($\langle\rangle$). The main difference between types and classes, besides the already mentioned different extension denotation, is that types denote acyclic complex structures, while classes denote complex structures where *cycles* are allowed. At instance level, we have, corresponding to types, *complex values* and, to classes, *abstract objects*. This distinction generalizes the usual one between the set of *base-values* $\mathcal{D}$ (i.e. objects typically hard wired as integers, characters, strings) and *abstract objects*, which must be explicitly introduced. An abstract object must be uniquely identified by an *object identifier* (*oid*) and has a description, which may change and is a *complex value*. A complex value is created by recursively applying to complex values and *oid*$_s$ any of the basic constructors: set ($\{\}$), tuple ($[]$), sequence($\langle\rangle$). Inheritance is represented, for types, by the *refinement relation* and, for primitive classes, by an explicit *isa relationship*.

The availability of classes with cyclic description, together with the idea of derived classes make the computation of extensions through an iterative process necessary in order to reach a fixpoint. Which type of fixpoint is more adequate is still matter of discussion: least, greatest, one in between. The choice adopted in [1, 2] is for a greatest fixpoint semantics. In fact, the greatest fixpoint tries to classify instances in the *most specific class* is compatible with the definitions, and thus seems to be more adequate in DB environment.

The idea developed in [1, 2] is to propose a terminological logic, allowing cycles, which captures the semantics of complex object models, thus permitting polynomial subsumption algorithms to guarantee consistency and minimality of a schema. The main extensions with respect to complex object models are: the *conjunction* operator which permits to express inheritance between classes(types) as a part of a class description and derived classes. In the following the formalization of the problem as proposed in [2] is shown.

Types and Values

We denote by $\mathbf{S}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{T})$ the set of all *finite type descriptions* $(S, S', \ldots)$, also briefly called *types*, over given $\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{T}$, which is defined as follows:

$$S \rightarrow B \mid P \mid D \mid T \mid \{S\} \mid \langle S\rangle \mid [a_1 : S_1, \ldots, a_k : S_k] \mid S \sqcap S' \mid \triangle S$$

where $\mathbf{A}$ is a countable set of *attributes* (denoted by $a_1, a_2, \ldots$), $\mathbf{B}$ a *PTIME base-type system*[1], $\mathbf{C}$ a countable set of *class names* partitioned in $\mathbf{C_p}$ and $\mathbf{C_d}$, where $\mathbf{C_p}$ is a set of *primitive classes* $(P, P', \ldots)$, $\mathbf{C_d}$ the set of *derived classes* $(D, D', \ldots)$, $\mathbf{T}$ a countable set of *type names* $(T, T', \ldots)$, such that $\mathbf{C}$, $\mathbf{T}$, and $\mathbf{B}$ are pairwise disjoint.

$\mathcal{O}$ being the set of *object identifiers* disjoint from $\mathcal{D}$, we can define the set of all *complex values over* $\mathcal{O}$, $\mathcal{V}(\mathcal{O})$, as the set of values obtained by finitely nesting elements of $\mathcal{O}$ and $\mathcal{D}$ with the constructors: $\{\}$, $[]$, $\langle\rangle$. We assign values to object identifiers by a total *value function* δ from $\mathcal{O}$ to $\mathcal{V}(\mathcal{O})$.

For a given set of object identifiers $\mathcal{O}$ and a value function δ, the *interpretation function* $\mathcal{I}$ is a function from $\mathbf{S}$ to $2^{\mathcal{V}(\mathcal{O})}$ such that:

$$\mathcal{I}[B] = \mathcal{I}_{\mathbf{B}}[B] \quad \mathcal{I}[C] \subseteq \mathcal{O} \quad \mathcal{I}[T] \subseteq \mathcal{V}(\mathcal{O}) - \mathcal{O}.$$

The interpretation of types is defined inductively for all $S, S' \in \mathbf{S}$ by the usual interpretation of the type constructor $\{\}$, $[]$, $\langle\rangle$, and as follows for $\sqcap$ and $\triangle$ operators:

$$\begin{aligned}
\mathcal{I}[S \sqcap S'] &= \mathcal{I}[S] \cap \mathcal{I}[S'] \\
\mathcal{I}[\triangle S] &= \left\{o \in \mathcal{O} \middle| \delta(o) \in \mathcal{I}[S]\right\} \\
\mathcal{I}[\top_C] &= \mathcal{O}.
\end{aligned}$$

This interpretation function is very general, but what we need is an interpretation of classes and types, consistent with their descriptions, denoted as *possible instance*, and a notion of *well-formed schema*. Further, the presence of cycles in class descriptions, lead to select a unique *legal instance* of a schema.

Database Schemata and Instances

We define a *schema* σ over $\mathbf{S}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{T})$ as a total function from $\mathbf{C} \bigcup \mathbf{T}$ to $\mathbf{S}$. We say that σ is *well-formed* if it is *type well-founded* (types defined using other type names always describe finitely nested values) and *inheritance well-founded.* Further, as inheritance is expressed as a conjunction, we can easily define direct inheritance as follows: $N \in \mathbf{C} \cup \mathbf{T}$ *inherits directly from* N', written $N \prec\!\!\!\mid N'$, iff N' compares as a conjunction term in $(\sigma(N))$. Thus, we can say that a schema is *inheritance well-founded* iff the transitive closure of $\prec\!\!\!\mid$, which is denoted by $\prec$, is a strict partial order.

We say that an interpretation function $\mathcal{I}$ as defined above is a *possible instance* of a schema σ iff the set $\mathcal{O}$ is *finite*, and for all $P \in \mathbf{C_p}, D \in \mathbf{C_d}, T \in \mathbf{T}$:

$$\begin{aligned}
\mathcal{I}[P] &\subseteq \mathcal{I}[\sigma(P)] \\
\mathcal{I}[D] &= \mathcal{I}[\sigma(D)] \\
\mathcal{I}[T] &= \mathcal{I}[\sigma(T)].
\end{aligned}$$

i.e.[1] a countable set of base-type designators which contains $\mathcal{D}$, is complete with respect to conjunction $\sqcap$ and such that $B \sqcap B'$ can be decided in polynomial time.

Now, we have to select among the possible instances, with identical $\mathcal{O}$ which share the same δ interpretation of primitive classes (denoted as $\mathbf{P}$), one instance, called *legal instance*, which is unique for derived classes, taking into account terminological cycles in class descriptions. We define as legal instance of a well-formed schema σ the unique *greatest* instance of $\mathbf{P}$.

Inheritance, Subsumption, and Coherence

We can now define a general *subsumption relation*, written $S \leq S'$ for $S, S' \in \mathbf{S}$ of a schema σ:

$$S \leq S' \text{ iff } \mathcal{I}[S] \subseteq \mathcal{I}[S'] \text{ for all legal instances } \mathcal{I} \text{ of } \sigma.$$

In [2] some interesting results on computational properties of *coherence* (absence of types and classes $\simeq \bot$) of a well-formed schema and subsumption computation are shown, specifying a mapping from schemata to nondeterministic finite automata. In particular, the two results of the following propositions are relevant to support minimality and consistency of a schema.

The coherence problem for a well-formed schema σ over $\mathbf{S}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{T})$ is in PTIME if the base-type system is binary compact.

Moreover, there is a wide class of database schemata which can be transformed into equivalent *normalized* schemata without an exponential increase for which subsumption computations can be done in polynomial time. A schema is called normalized iff

- for all $S \in \mathbf{S}$, $\sigma(T)$ does not contain any conjunction, and
- for all $C \in \mathbf{C}$, $\sigma(C) = \sqcap P_i \sqcap \triangle S$ such that S does not contain any conjunction, $P_i \in \mathbf{C_P}$ and $\triangle S \leq P_i$, for all i.

The subsumption problem for normalized schemata is in PTIME.

References

[1] D. Beneventano, S. Bergamaschi, and C. Sartori. Taxonomic reasoning in LOGIDATA$^+$. In V. Monaco and R. Negrini, editors, *COMP-EURO 91*, pages 894–899, IEEE Computer Society Press, Bologna - Italy, May 1991.

[2] S. Bergamaschi and B. Nebel. *Theoretical Foundations of Complex Object Data Models.* Technical Report 74, CIOC - CNR, Bologna - Italy, Dec. 1990.

[3] S. Bergamaschi and C. Sartori. *On taxonomic reasoning in conceptual design.* Technical Report 68, CIOC - CNR, Bologna - Italia, March 1990.

[4] D. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *IJCAI 91*, Australia, 1991.

[5] R. Hull and R. King. Semantic database modelling: survey, applications and research issues. *ACM Comput. Surv.*, 19(3):201–252, Sep. 1987.

[1]systems such that for each subset $X \subseteq \mathbf{B}$ with $\sqcap X = B$, there are two elements $B', B'' \in X$ such that $B' \sqcap B'' = B$

Interests and Issues in Description (Terminological) Logics

Ronald J. Brachman
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974
U. S. A.

1 Current Research Interests

My current work on knowledge representation spans several research topics:

CLASSIC. My primary focus has been on the CLASSIC Knowledge Representation System [1], which we have been developing at Bell Labs over the course of the last several years. CLASSIC is a relatively small system intended for use in a limited number of applications; we do not intend it as an all-purpose tool usable in any conceivable application (nothing prevents that, it simply doesn't have enough power to be of much use in certain high-powered applications, such as natural language understanding). However, we have worked hard to make CLASSIC understandable to less-than-expert users; one of our goals is to allow this kind of KR technology to be used by non-AI people, and many of our design decisions are based on the need for it to be straightforward and easily learnable by those not well-versed in knowledge representation research or philosophy. As a result, we have kept the representation language uniformly compositional and relatively simple.

We have completed the design and COMMON LISP implementation of CLASSIC 1.0, which has now been reimplemented in C by a Development organization and used in a significant application within Bell Labs, as well as in several courses on knowledge representation (e.g., University of Pittsburgh, Columbia University). Based on feedback from users and applications, we have completed the design of a number of extensions to CLASSIC (a role hierarchy, role inverses, more useful rules, etc.), and have begun the implementation of the second generation of the system. When we are finished with CLASSIC 2.0, we believe we will have a very usable and reasonably expressive system, and do not have plans to keep expanding the representation language. Our main focus will be on providing tutorial information for new users, making the system more usable in real application situations, and building applications ourselves. While we hope to stabilize the representation language, we will need to handle connections with databases, more elaborate querying facilities, explanation of reasoning, persistence of CLASSIC knowledge bases, and possibly extensibility. We are also currently well along in the implementation of a graphical interface.

Extensibility. Alex Borgida and I have been working on an architecture that would make classification-based systems extensible in an interesting and efficient way. We have modularized and abstracted the general structure of reasoning systems that do classification, and have specified a set of functions that the user can provide to extend the language to handle new constructs (these functions include things like parsing, determining inconsistency between two constructs of the same type, determining a set of other constructs implied by a new one, etc.). Alex has been fleshing out the complete specification of these functions, the hooks that need to be placed in the system to allow them to be invoked at the right time, and some example novel constructs that can be created and integrated with this mechanism. With this approach, the basic system can be small and compact, and can be extended to different versions without obligating a user to take on the overhead of constructs not useful in his application—and without redesigning or reimplementing any of the core of the basic system.

CLASSIC as a Data Model. We are investigating the potential use of CLASSIC as, roughly speaking, a semantic data model, allowing a user to impose a more complex, object-centered worldview over data stored in a (or many) relational database(s). Peter Selfridge has built a prototype system [2] in which the user takes a simple relational model of cross-reference relations in software (i.e., what function calls what function) and, with CLASSIC essentially extends the schema as he discovers interesting facts and relations in the code. We are also exploring a more ambitious version of the same idea in a more business-oriented domain. One thing has become clear in considering these applications—we need a serious query language with which to interact with the combined knowledge/database. We have designed a set-oriented query facility that integrates CLASSIC expressions in a simple manner, does enough of what SQL does to satisfy our users (who are used to SQL over INGRES databases), and does some things that the users want but which cannot be done in SQL. A query processor for this language has been implemented (by Tom Kirk), and we are now experimenting with it. We have also considered the issues of persistence in CLASSIC knowledge bases, as well as various modes of connection with standard databases.

2 Discussion Topics

The topics that I believe are most critical to be discussed by our community are closely tied to these current research interests, as well as to some general concerns about knowledge representation work in general, and how we can better address the needs of real users of our systems. Here I will break these interests down into three categories: theoretical issues, systems issues, and "meta"-issues. I addressed some of these in my keynote talk, and several were discussed at the workshop.

Theoretical Issues. In the last few years, we have been besieged by complexity results for various forms of terminological systems. While these results continue to be useful, they don't give us much general insight into how to deal with normal uses of terminological systems, and when worst cases might arise.[1] It is time to turn our attention to concrete

[1]I should say that the recent comprehensive results presented by Donini, Nutt, et al., at the workshop and at KR'91 are both impressive and insightful. They consolidate many of the individual results we have

patterns of use, and real styles of knowledge base, and understand what are the sources of computational complexity that will really affect our use of TL systems in practice. I think that we still need formal results, and we should avoid handwaving of the "well, it's never arisen in my experience" sort. But our attention should be focused on "normal" cases, under certain sets of well-specified assumptions (what these are will need to be fleshed out, of course). Ideas that come to mind include consideration of limited-depth definitions, trees of certain shapes, complexity measures for individual term structures (i.e., how complex is a given concept?), etc.. Also, if we look more at the detailed algorithmic complexity (rather than just computability or NP-hardness) of some of our algorithms, we can see in more detail what parts of the input the complexity hinges on. This should yield a level of insight not yet achieved with these systems.

Other issues need careful attention on the theoretical side:

- sequence and ordering: we have considered some domains where dates and other ordering functions play a crucial role. How can we fold partial (and total) orders into our standard TL systems?
- "structural descriptions": this was a critical piece of the original KL-ONE proposals; it is probably time to resurrect it and nail down some formal proposals, with their semantics and some understanding of their contribution to overall complexity.
- metaclasses: we have begun to run into cases where metaclasses are important. It probably is not too difficult to fold such things into our basic TL architecture, but formal work needs to be done to get this straight. The issue of such classes seems to be arising in practice now: Bill Swartout presented his needs at the workshop, Alfred Kobsa talked about reification in SB-ONE, and we have seen the need for at least aggregate class information, such as average values of certain roles, in one of our applications.
- query languages: in many applications where a TL-based system operates like or with a database, it is impossible to get along with the common lack of languages designed for querying. One can imagine extremely interesting query languages that combine the best of standard relational languages, languages like SETL, and the object-oriented contribution of our term-forming operators. We have begun extensive work on such a language, and the workshop indicated that the problem has arisen for others as well.
- relation of TL's to type theory: much work in programming languages and databases involves complex types that bear a great deal of similarity to our terms. I would like to see a detailed analysis of the commonalities and the differences of these approaches.
- on the encouraging side, recent work by Smolka, et al., reported at the workshop, shows that important connections to feature logics and constraint logic programming seem poised to bear fruit.

seen over the last few years, and provide important information on what the root causes of complexity in these languages are.

Systems Issues. There are many issues that we should be concerned with that involve the utility of TL-based systems in the real world. I would like to see serious attention given to the issues that arise in trying to use our systems in applications. These might include database/knowledge base integration, extensible TL systems, and discussion of various types of ABoxes and their integration with the TL system. We might want to consider alternate forms of classification (i.e., less purely deductive forms). We might also consider the desirability of publication of detailed algorithms for subsumption, classification, propagation, etc., so that everyone can benefit from advances at the systems level made by the numerous projects implementing TL systems. I think we all agree that if we can find the right place (probably a journal) to publish such algorithms, the entire community would benefit.

"Meta"-issues. There are a variety of issues that are not particularly technical, but are worthy of discussion by our research community. For one thing, I think that our little community is thought of as a perhaps small "faction" within the KR community. How can we make it clear that our work is not as limited as we sometimes make it appear, and that it brings something to the representation task that is complementary to other more well-known approaches (i.e., classification)? I think we should make some effort to avoid being viewed as a small group of people overly concerned with complexity results in severely limited, highly technical formalisms. For my part, in line with discussion at the workshop, I plan to write a paper to submit to, say, *The AI Magazine*, in which I intend to point out the impressive breadth of interests and accomplishments represented at the Dagstuhl workshop.

Other issues of a similar nature that come to mind are these:

- What should the stance of our community be with respect to the recent efforts towards "standards" in knowledge representation? Several people attending this workshop have been playing key roles in DARPA-sponsored work on sharing knowledge bases. Should we as a community take a stand? Should we become more active? Less active? For now, it looks like many of us will remain active in the "knowledge representation system specification" subpart of the community-wide effort.

- How do we evaluate work in our area? This applies to both our own work, as well as to the broader KR community at large.

- There is the ever-lingering issue of how expressive the languages in our systems should be. I personally think we need to start thinking about the set of ways a knowledge representation system can be—there is a continuum of types of systems, ranging from relatively inexpressive to extremely expressive, and that there simply is no way to state a set of criteria (with respect to expressiveness) that should necessarily apply uniformly to *all* KR systems. Limited systems have their places, and have been quite successful at certain tasks. More expressive systems are needed for language applications of certain sorts, perhaps some medical applications as asserted by Doyle and Patil, and others. I would like to see us straighten this out once and for all, so we can stop the tendency to make global, context-free pronouncements on the adequacy of certain KR systems. We should begin to examine the trade-offs that affect the character of KR systems, and see what useful points lie on what is clearly a continuum. I must say that good progress on clearing this up was made at the workshop, thanks in part to Bob MacGregor and Bill Swartout.

- Finally, I think it is quite important for us as a community to begin to characterize in some detail the types of applications for which classification- or description-based systems have been and could be successful. Experience teaching with CLASSIC, for example, shows that students keep asking, "what is this good for?" I think we have plenty of instances of success that we can point to. I would like to see a compendium of successful uses, so that we can show the broader community what they are good for, and how they complement other things in wider use. Having a workshop on the topic this Fall (in Berlin) is a great step in the right direction.

3 What's in a Name?

I would like to make one final comment, with respect to the description of the type of system we are all engaged in studying. While "terminological logic" has its appeal, and is representative of part of what we do, it seems not only too complicated to say, but too narrow to characterize our community as a whole. Similarly, "KL-ONE-like systems" has a ring of truth to it, but is too limiting. Personally, I believe that the word *description* should play a central role in how we present what we are doing. While *description logics* is still not an ideal label (in part because it seems to virtually leave out work on assertional components), it seems much better to me than the others. Among other things, it broadens our community to include interaction with others doing related work, but not sharing our literal ancestry (e.g., OMEGA). Since so much of what we do is actually about forming, relating, and using descriptions, I think we should give this new term a try and see how it works.

References

[1] R. J. Brachman, A. Borgida, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Resnick. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, California, 1991.

[2] P. G. Selfridge. Knowledge representation support for a software information system. In *Proceedings of the Seventh IEEE Conference on AI Applications*, pages 134–140. Miami Beach, Florida, Feb. 1991.

Intensional Semantics and Relationships between Epistemology and Ontology

Amedeo Cappelli
Istituto di Linguistica Computazionale - CNR
Via della Faggiola, 32 56100 Pisa Italy
Phone 39 50 560481
Fax 39 50 589055
e-mail SISTEMI@ICNUCEVM.CNUCE.CNR.IT

1 Introduction

Many problems connected with the term subsumption language paradigm have so far not received a solution. Furthermore, many interesting issues, suggested at the very beginning of the history of this subject, havebeen undecided, for various reasons (Brachman & Shmoltze, 1985). One of the major assumptions in designing knowledge representation formalisms in the KL-One family, was the so-called "intensional representation" introduced by Woods (1975) and Brachman (1979). An intensional representation is required when two descriptions have to be compared (Bobrow & Winograd, 1977), or when they are interpreted by qualitative processes; in other words, many processes can be activated by using the global structure of a concept, and by interpreting its pro perties and the relationships between these properties (Woods, 1990). Is this a problem which goes beyond the actual goals of a language, in the sense that it lies in the realm of knowledge and is domain-dependent, or can it be approached in a very general way, by defining a formal semantics of the possible intensional operations which can be specified on a terminology? The classic problem of "structural descriptions" has to be seen in this perspective, as well. It is evident that an adequate representation of a concept involves the specification of the relationships existing between its descriptive parts. Structural description was one of the most interesting data structures of the classic SI-Nets model, even if it may be considered as lying in an unclear position between the conceptual and the epistemological levels, with reference to Brachman's distinction (1979). Another problem, which is still open, is that of the role, of its function and of its meaning. Once, it was considered as a locus where many types of processes may be accumulated, such as cardinality and modality. Apart from the problems which arise when modality and number are interpreted together, the role is a complex link which makes it possible for concepts to interact with one another. So, while in certain cases it can be seen as a mere tuple, such as, for instance, in data base application of terminological languages, in other cases it must be seen as the point where several conceptual processes take place, such as, for instance, in expert systems or in other knowledge-based applications (Cappelli et al, 1983a, Cappelli & Moretti, 1983b, Cappelli et al., 1986, Cappelli et al. 1988, Cappelli, 1987, Caracoglia, 1988, Caforio, 1988). As to the former, it is sufficient to verify the tuple in the extension of a concept, whereas in the latter, it is relevant to interpret the association from a well-defined conceptual point of view. In this way, the ontological meaning of a role plays an important role, as suggested in Frederking & Gehrke, 1988, Winston et al., 1987). Ontology plays an important role in structuring knowledge. In order to create a knowledge base, one must make some assumptions about what kinds of things there are in the world; in other words, any user needs a general grammar for representing knowledge - in the sense of Brachman (1979), a notion which has been lost, as claimed in Doyle & Patil (1989), but he must also be guided by using constraints depending on the nature of the things being modelled (Lenat & Guha, 1990, Niremburg & Monarch, 1987). This limits the generative power of the grammar, but, in any case, its expressive power increases, since putting together an epistemological formalism and a set of ontological constraints makes it possible to account for more subtle conceptual facts.

2 Intensional semantics

An intensional semantics for a typical terminological language has been designed (Mazzeranghi & Cappelli, 1990), which is quite different from the extensional models so far proposed (Brachman et al. 1985, Patel-Schneider, 1989, Shmoltze, 1989, Nebel, 1988, 1990). The semantics of the language is similar to that of data types in programming languages. Primitive concepts are denoted by a set of values. Defined concepts are denoted by their properties. A denotation thus contains the minimum number of properties which are required for an individual to be an instance of a generic concept. More precisely, the denotation is the Cartesian product of the sets denoting the properties of the generic concept (deduced from its syntax). A role is denoted by a function which, given a tuple, returns the values of the property which individuates the role. In general, in any hybrid system, the assertional component is procedural, since it allows a user to make assertions about the individual concepts (creation of individuals, link of individuals by roles, etc.), by updating an assertional knowledge base. This approach has certain consequences, such as, for instance, that a user has to know the entire history of the KB, since any individual is characterized by the entire sequence of declarations, and no constraints exist in order to control the use of the KB, so that the probability for a KB to become inconsistent increases with its dimension. Different results can be achieved if a functional approach based on the intensional semantics is adopted. An individual is created by instantiating the properties of the relative generic concept, used as a guide. As a result, this process creates a tuple whose elements are the instantiating properties. The instantiation chain terminates by instantiating primitive concepts on the basis of their denotation. Certain extensions have been introduced into the assertional component in order to make it possible to use it more easily way, such as the following:

- Declaration of individuals
- Properties of individuals specified by using their names
- Properties of individuals specified incrementally

A deeper integration between the entire system and any programming language can be reached, which implicitly gives rise to an object-oriented system. In other words, it is possible to introduce an individual concept into a programming language, like any other data type. For instance, an individual concept is passed to a function as a parameter; once it has been verified that this individual is an instance of a generic concept, or of one of its subconcepts, the function will be executed. For this aim, it is required that the identifiers of the programming language can be used as identifiers of individual concepts. The system has to expand the identifiers with their definitions which have been evaluated in a different environment. Consequently, the system can assume different behaviours: functional or procedural in accordance with the use of the programmming language identifiers. Furthermore, an increase in the expressive power is obtained, since it makes it possible to give a formal meaning to all kinds of recursive definitions of individual and generic concepts. For instance, by using an indirect recursion, the concepts of husband and wife can be defined as follows:

```
HUSBAND=(and  HUMAN-BEING
    (all  wife  WIFE)
    (atleast  1  wife)
    (atmost  1  wife))

WIFE=(and  HUMAN-BEING
       (all  husband  HUSBAND)
       (atleast  1  husband)
       (atmost  1  husband))).
```

3 Structural descriptions

Structural descriptions are considered as an object-oriented programming tool. More precisely, a function or a procedure can use the roles of a describing concept in order to refer to the roles of a described concept. For instance, a function can be written in order to calculate the height of any object composed of two parts, one of which is on the other, simply referring to the roles "is-on" and "is-under" of a previously defined concept "ON". This function can be applied to any individual, whose generic concept has been previously defined by using the concept "ON", such as, for instance a TABLE, which is composed of a board and four legs, or a HOUSE, which is composed of a roof and certain walls. A new form of inheritance by structural descriptions is thus realized.

4 Integration between epistemology and ontology

The properties of a concept play a relevant role from an intensional viewpoint, in the same way as types of concepts are essential if we look at the universe as a map of complex descriptions interacting one with the other. Such facts can be specified by using notions such as, for instance, sortal concepts, or natural, nominal and artifact concepts as defined in the psychological paradigm (Cappelli & Catarsi, 1990; Keil, 1979, 1989; Smith & Medin, 1981; Wiggins, 1980), or ontological notions, such as substances and accidents, genus, eidos, etc (Simons, 1983). Experiments about the relationships between epistemology and ontology are now being carried out, in the aim of both investigating the ontological adequacy of certain SI-Nets data structures and integrating epistemological tools with ontological constraints. Classical SI-Nets formalism accounts for many ontological facts, for instance the representation of hierarchies and properties of concepts can easily be translated into concepts, roles and cables. However, descriptions of objects need to be further specified from a conceptual point of view, in particular when a description is given in the aim of specifying global constraints which control the application of intensional functions A constructive formal apparatus is thus realized in which epistemology, a la Brachman, and ontology are integrated. A system has been created in which the representational tools based on intensional semantics interact with an ontological representation of a portion of universe; in this way, a user can create a knowledge base by using this representation as a guide, imposing constraints on the descriptions of items and their insertion into the network.

References

- Bobrow D. G., Winograd T., An Overview of KRL, a Knowledge Representation Language, Cognitive Science, 1 (1977).
- Brachman R. J., On the epistemological status of semantic networks, in N. Findler (ed.), Associative Networks: Representation and Use of Knowledge by Computers, New York: Academic Press, 1979: 3-50.
- Brachman R. J., Fikes R. E., Levesque H. J., An essential hybrid reasoning system: knowledge and symbol level accounts in KRYPTON, in Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los Angeles (Ca): Kaufmann,1985: 532-539.
- Brachman R. J., Schmolze J. G., An overview of the KL-ONE Knowledge Representation System, Cognitive Science 9 (1985).
- Cappelli A.,Catarsi M. N. , The Role of Ontology in Structuring Knowledge, ILC-KRS-1990-4, Pisa, 1990.
- Cappelli A., Moretti L., Vinchesi C., KL-Conc: a Language for Interacting with an SI-Net, in Proceedings of the 8th-IJCAI Conference, Los Altos: Kaufmann, 1983.
- Cappelli A., Moretti L., An Approach to Natural Language n the SI-Nets Paradigm, in Proceedings of First Conference of ACL-Europe, Pisa, 1983b.
- Cappelli A., Caracoglia G., Moretti L., A Chunking Mechanism for a Knowledge Representation System, Cybernetics and Systems, 17 (1986) pp. 277-287.
- Cappelli A., Semantic Networks and Natural Language Understanding, in "Research and Development in Language Processing" Paris December 7-11, 1987.

- Cappelli A., Moretti L., Pagni F., Verso la costruzione di una base di conoscenza per un sistema di aiuto ad un esperto in radioprotezione, in Atti del Congresso Internazionale "Informatica e regolamentazioni giuridiche", Roma, 1988.

- Caracoglia G., Skill Acquisition in a Knowledge Representation System, in Proceedings of the Third International Symposium on Knowledge Engineering, Madrid, 1988.

- Caforio M., A Network Search Approach Based on the Chunking Mechanism, in Proceedings of the Third International Symposium on Knowledge Engineering, Madrid, 1988.

- Doyle J., Patil R. S., Two Dogmas of Knowledge Representation: Language Restrictions, Taxonomic Classification, and the Utility of Representation Services, MIT/LCS/TM-387.B, Cambridge (Mass.), 1989.

- Frederking R. E., Gehrke M., Resolving Anaphoric References in a DRT-based Dialogue System, in H.Trost (ed.), 4 Osterreichische Artificial-Intelligence-Tagung, Springer,1988, 94-103.

- Hobbs J.R., Croft W., Davies T., Edwards D., Laws K., Commonsense Metaphysics and Lexical Semantics, Computational Linguistics 13 (1987).

- Keil F. C., Semantic and conceptual development, Cambridge (Ma.): Harvard University Press, 1979.

- Keil F. C., Concepts, Kinds, and Cognitive Development, Cambridge: MIT Press, 1989.

- Lenat D. B., Guha R. V., Building Large Knowledge-Based Systems, Representation and Inference in the Cyc Project, Reading (Ma.): Addison-Wesley, 1990.

- Mazzeranghi D., Cappelli A. An Intensional Semantics for a Terminological Language, ILC-KRS-1990-5, Pisa, 1990 .

- Nebel Bernhard., Computational Complexity of terminological reasoning in BACK, Artificial Intelligence, 34 3 (1988): 371-383.

- Nebel Bernhard., Reasoning and R evision in Hybrid Representation Systems, Berlin: Springer Verlag,1990.

- Niremburg S., Monarch I., The role of Ontology in Concept Acquisition for Knowledge-Based Systems, Carnegie- Mellon University, Pittsburgh, PA, 1987.

- Patel-Schneider Peter F., Undecidability of subsumption in NIKL, Artificial Intelligence, 39 2 (1989): 263-272.

- Schmolze James G., The Language and Semantics of NIKL, Technical Report 89-4, Department of Computer Science, Tufts University, Medford, Mass., September 1989.

- Simons P., A Lesniewskian Language for the Nominalistic Theory of Substance and Accident, Topoi 2 (1983): 99-109.
- Smith E. E., Medin D. L., Categories and Concepts, Cambridge (Mass.): Harvard Univ. Press, 1981.
- Wiggins D., Sameness and Substance, Oxford: Basil Blackwell, 1980.
- Winston M. E., Chaffin R., Herrmann D., A Taxonomy of Part-Whole Relations, Cognitive Science 11 (1987), 417-444.
- Woods W. A., What's in a link: foundations for semantic networks, in Bobrow and Collins (eds.), Representation and Understanding: Studies in Cognitive Science, New York: Academic Press, 1975: 35-82.
- Woods W. A., Understanding Subsumption and Taxonomy: A Framework for Progress, TR-19-90, Harvard Univ. Center for Research in Copmputing Technology, Cambridge (Mass.) 1990.

Tractable Concept Languages

Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi
Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza"
via Salaria 113, I-00198, Roma, Italy
e-mail: {donini,lenzerini,nardi}@vaxrma.infn.it

Werner Nutt
Deutsches Forschungszentrum für Künstliche Intelligenz
Postfach 2080, D-6750 Kaiserslautern, Germany
e-mail: nutt@dfki.uni-kl.de

Concept languages provide a means for expressing knowledge about hierarchies of concepts, i.e. classes of objects with common properties. They have been investigated following the ideas initially embedded in many frame-based and semantic-network-based languages, especially the KL-ONE language [2]. In contrast to earlier formalisms, concept languages are given a Tarski style declarative semantics that allows them to be conceived as sublanguages of predicate logic [8].

The basic reasoning tasks on concepts are unsatisfiability and subsumption checking. A concept is unsatisfiable if it always denotes an empty set. A concept C is subsumed by a concept D if C denotes always a subset of D. Since the performance of any application developed using concept languages will heavily rely on the above reasoning tasks, it is important both to characterize their computational complexity and to devise algorithms as much efficient as possible.

Recent results allow us to draw a fairly complete picture of the complexity of a wide class of concept languages [3, 4, 9]. Such results have been obtained by exploiting a general technique for satisfiability checking in concept languages. The technique relies on a form of tableaux calculus, and has been proved extremely useful for studying both the correctness and the complexity of the algorithms.

The work reported here is concerned with the design of concept languages including the most powerful set of constructs, while retaining the tractability of subsumption, in particular extending the basic polynomial language $\mathcal{FL}^-$ [1]. If C and D denote generic concepts of $\mathcal{FL}^-$, and R denotes a role, $\mathcal{FL}^-$ includes the following constructs:

$$\mathcal{FL}^- : \left\{ \begin{array}{cl} C \sqcap D & \text{conjunction of concepts} \\ \forall R.C & \text{universal role quantification} \\ \exists R & \text{unqualified existential role quantification} \end{array} \right.$$

Various extensions of $\mathcal{FL}^-$ with a polynomial subsumption problem have already been considered:

- $\mathcal{FL}^-$ + role concatenation $R \circ Q$ (also called role chaining, see [1]);
- $\mathcal{FL}^-$ + concept formed by imposing number restrictions on roles $(\geq n\, R)$, $(\leq n\, R)$ (see [7]);
- $\mathcal{FL}^-$ + role conjunction $R \sqcap Q$ (see [7]);
- $\mathcal{FL}^-$ + negation of primitive concepts $\neg A$ (see [9]).

We considered concept languages obtained by combining constructs chosen from the ones presented till now, plus the following ones: union $C \sqcup D$, qualified existential role quantification $\exists R.C$, negation of general concepts $\neg C$, inverse roles R^{-1}. We did not consider any syntactic restriction on the possible combinations of the chosen constructs—i.e. we considered only fully compositional concept languages.

The result of our work is the definition of two new extensions of $\mathcal{FL}^-$, called $\mathcal{PL}_1$ and $\mathcal{PL}_2$. We show that subsumption in both languages can be solved in polynomial time. Moreover, they are maximally expressive, in the sense that none of the constructs previously considered can be added to them without losing tractability. It is interesting to notice that both languages include the construct for inverse roles, which has not been considered up to now in tractable languages.

In particular, $\mathcal{PL}_1$ extends $\mathcal{FL}^-$ in the following way:

$$\mathcal{PL}_1 : \begin{cases} C \sqcap D, \forall R.C, \exists R & \text{(the language } \mathcal{FL}^-\text{)} \\ \neg A & \text{negation of primitive concepts} \\ (\geq n\, R), (\leq n\, R) & \text{number restrictions} \\ R^{-1} & \text{inverse roles} \end{cases}$$

$\mathcal{PL}_1$ can be therefore considered maximally expressive relative to the costructs available for concepts.

On the other hand, $\mathcal{PL}_2$ extends $\mathcal{FL}^-$ as follows:

$$\mathcal{PL}_2 : \begin{cases} C \sqcap D, \forall R.C, \exists R & \text{(the language } \mathcal{FL}^-\text{)} \\ R \sqcap Q & \text{role conjunction} \\ R \circ Q & \text{role concatenation} \\ R^{-1} & \text{inverse roles} \end{cases}$$

$\mathcal{PL}_2$ can be therefore considered maximally expressive relative to the costructs available for roles. For a detailed description of $\mathcal{PL}_1$ and $\mathcal{PL}_2$ see [5].

The question arises about how many other maximally expressive tractable languages can be obtained by extending $\mathcal{FL}^-$ with the above constructs. With regard to this point, we can state an interesting property of the two languages proposed: *let L be a concept language extending $\mathcal{FL}^-$ with any combination of the constructs presented above; if the subsumption problem in L is tractable, then the set of constructs of L is either a subset of those of $\mathcal{PL}_1$ or a subset of those of $\mathcal{PL}_2$. There is only one exception to this statement, namely the language extending $\mathcal{FL}^-$ with both role chaining and number restrictions.* This exception is currently under investigation.

As a conclusion, we want to comment on the results of the research on the computational properties of concept languages by means of the satisfiability checking technique.

We think that the outcomes of this body of research go far beyond a mere complexity analysis. In particular, they shed light on three basic aspects related to the use of concept languages in knowledge representation.

- First of all, since the complexity of both satisfiability and subsumption depends upon the constructs allowed in the language, they provide a useful framework for the study of the trade-off between the expressive power of the languages and their inherent complexity, which was the initial motivation of the seminal work by Brachman and Levesque [6].
- Secondly, the design of concept languages can now be realized through the application of the above mentioned technique, which provides an algorithmic framework that is parametric with respect to the language constructs.
- Thirdly, the study of the computational behaviour of concept languages has led to a clear understanding of the properties of the language constructs and their interaction. This knowledge about the structure of concept languages can thus be used in the design of intelligent reasoning procedures, that—by looking at the form of concepts—can reason about the deductive service, for example estimating the difficulty of performing the required deduction, attempting to provide quick answers to subproblems, or trying possible simplifications of the problem.

References

[1] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence AAAI-84*, 1984.

[2] R. J. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[3] F. M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz, Postfach 2080, D-6750 Kaiserslautern, Germany, 1990.

[4] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning KR-91*. Morgan Kaufmann, 1991.

[5] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence IJCAI-91*, Sidney, 1991.

[6] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[7] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988.

[8] B. Nebel and G. Smolka. Representation and reasoning with attributive descriptions. In K. Bläsius, U. Hedtstück, and C.-R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, number 422 in Lecture Notes in Artificial Intelligence, pages 112–139. Springer Verlag, 1990.

[9] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

Extending Hybridity within the YAK Knowledge Representation System

Enrico Franconi
IRST
I-38050 Povo TN, Italy
e-mail: franconi@irst.it

1 YAK

YAK [6] is a hybrid KR system, and in its foundations is similar to Classic [3] and Loom [8]. The core of the system is a "traditional" TBox/ABox hybrid representation language (with some peculiarities), enhanced, possibly in a "principled" fashion, with other hybrid modules representing different kind of knowledge and reasoning. The system, fully implemented in CommonLisp (and with an optional graphical user-interface machine-dependent), is the main knowledge representation module of the ALFresco natural language system, a multimodal dialogue prototype for the exploration of Italian art history.

The expressivity of the YAK TBox comes out from a study about the balancing of expressiveness, functional adequacy and formal properties of deductive procedures. It has been shown in [5] how some constructs — though enhancing expressive power and still maintaining tractability — do not give an intuitive behaviour to the language, from the point of view of both meaning and calculus.

The classifier provided within the YAK system has a tractable, sound and complete algorithm. Moreover the classifier is speeded up by a caching mechanism and an intelligent name expansion scheduling, borrowed from the tabular approach used in nondeterministic natural language parsers.

The ABox is a simple object oriented language, just like Classic: individuals are instances of concepts and incomplete descriptions are allowed. The query language has the same expressive power of the individual description language, with one variable.

2 Extending Hybridity

I quote James Schmolze [11] because I believe that this position is still of great topical interest:

> Terminological systems must fit within larger representational frameworks. It is therefore time to assess how well previous and current KR systems have integrated terminological, (grounded) propositional, equational, rule- based

and other representations and reasoners. (...) it is important to address the larger context in which such systems are placed and used.

Motivations and new ideas for the KR field often are originated within the natural language processing community. Prototypical knowledge for prediction in natural language understanding [7], belief representation for user modeling in a multi-agent dialog [2], reasoning about sets to handle conjunctions, plurals and natural quantifiers [1], reasoning about time relations [12] [10] are some aspects that we have taken into consideration. We are also exploring the possibility of reasoning in a coherent way with procedural attachments (following Classic) and production rules [14], and of using Weyhrauch's FOL system as an alternative ABox [4].

Within the framework of a complex hybrid architecture supporting multiple reasoning modalities, several aspects must still be addressed. The need of a greater expressivity to represent complex relations in natural language should be considered a major topic in the KR research — see, as an example, the KODIAK system [13]. Another important issue concerns the inference control procedure in the assertional component, as far as a single conclusion can arise out of different modalities which have different import. In these cases a belief revision mechanism [9] is central in order to manage nonmonotonic effects.

2.1 Prototypes

In this research project [7] the problem of instance recognition within an extended hybrid knowledge representation system is addressed. Structural aspects of concepts are represented at two separate levels, the terminological and the prototypical; individuals are expressed in the frame-based assertional component. The hybrid reasoning mechanism recognizes the type of the individuals with respect to the terminology, making use of reasoning with prototypes.

Basic ideas are shared with the so called Dual Theory about the mental representation of concepts. Within this theory concepts have a twofold representation: a "core description", useful for compositional meaning, and an "identification procedure" for typical instance recognition. Our own realization of such a distinction is that the core strictly defines the necessary and sufficient properties for the concepts (only the necessary ones in the case of primitive concepts), while the identification procedure is a similarity mechanism that works over a collection of perceptual and functional properties. We call such a collection the *prototype* for that concept. Within the identification procedure a "similarity model" is introduced that describes the probability rating that an object belongs to a class, supported by the similarity that the object shares with the prototype of that class.

The hybrid reasoning mechanism we propose extends the recognizing process of individuals in the assertional component. It makes use of the terminological knowledge to derive a first type assignment for the individual. This attribution is successively improved by comparing the description of the individual (via the similarity mechanism) to prototypes stored in the prototypical component. Prototypical knowledge is linked to appropriate names in the terminology through primitive concepts.

The apparatus distinguishes between qualitatively different information and yet can deal with the problem of preferences among the results of similarity-based reasoning.

2.2 Beliefs

This work is the attempt to import into the hybrid framework the ideas about *relevant beliefs* of [2]. The goal is to model an artificial agent — the system — which reasons subjectively about the beliefs of other agents in communication with him, in addition to its own beliefs.

The knowledge base has been partitioned into *viewpoints* each one representing a set of complex nested beliefs, i.e. what the system believes the agent `A` believes the agent `B` believes ... about some topic. Topics are simply individual descriptions (or, more generally, ABox propositions) present in the viewpoint. A topic is *believed with respect a viewpoint* if it is logically implied by the knowledge directly stated in the viewpoint or if it is entailed by *ascription*. The ascription mechanism tests the truth value in the "preceding" viewpoints according to a "particular" order; the process fails if at some point a contradiction is detected. The *relevant beliefs* theory presents a method concerning the ascription mechanism for determining whose beliefs are relevant in generating nested beliefs and in what order are they relevant.

Within YAK a 3-values — true, false, unknown — hybrid retrieval function (`believe?`) has been implemented, which recursively checks the logical implications and the contradictions in the nested viewpoints.

References

[1] Jürgen Allgayer. SB-ONE$^+$ — dealing with sets efficiently. Proceedings of *9th European Conference on Artificial Intelligence*, Stockholm, Sweden (1990)

[2] Afzal Ballim and Yorick Wilks. Relevant Beliefs. Proceedings of *9th European Conference on Artificial Intelligence*, Stockholm, Sweden (1990)

[3] Ronald J. Brachman, Deborah L. McGuiness, Peter F. Patel-Schneider, Lori Alperin Resnick and Alexander Borgida. Living with CLASSIC: When and How to Use a KL-ONE-Like Language. In J. Sowa (ed.) *Principles of Semantic Networks*, Morgan Kaufmann (1991)

[4] Paolo Bresciani. Logical Account of a Terminological Tool. Proceedings of *Applications of Artificial Intelligence IX*, Orlando FL (1991)

[5] Roldano Cattoni and Enrico Franconi. Walking through the Semantics of Frame-Based Description Languages: a case study. In Z.W. Ras, M. Zemankova, M.L. Emrich (eds.), *Methodologies for Intelligent Systems, 5*, North-Holland (1990)

[6] Enrico Franconi. The YAK (Yet Another Krapfen) manual. *IRST - Manual 9003-01*, Trento, Italy (1990). Also as *Progetto Finalizzato CNR* 'Sist. Informatici e Calcolo Parallelo' *report 7/30* (1990)

[7] Enrico Franconi, Bernardo Magnini and Oliviero Stock. Prototypes in a Hybrid Language with Primitive Description. To appear in *Computer & Mathematics with Applications, special issue: Semantic Networks in Artificial Intelligence*, Pergamon Press (1991)

[8] Robert MacGregor, A Deductive Pattern Matcher. Proceedings of *AAAI-88*, St.Paul MINN, 403-408 (1988)

[9] Bernhard Nebel, *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Artificial Intelligence 422, Springer-Verlag (1990)

[10] Albrecht Schmiedel. A Temporal Terminological Logic. Proceedings of *AAAI-90*, Boston MA (1990)

[11] James G. Schmolze. Statement of Interest for the *Workshop on Term Subsumption Languages*, Thorn Hill NH (1989)

[12] Paolo Terenziani, Pietro Torasso, Luisa Farinasso and Laura Mantegazza. Causation and Time in a Hybrid Knowledge Representation Formalism. In Z.W. Ras, M. Zemankova, M.L. Emrich (eds.), *Methodologies for Intelligent Systems, 5*, North-Holland (1990)

[13] Robert Wilensky. Some Problems and Proposals for Knowledge Representation. *Report No. UCB/CSD 87/351*, University of California, Berkeley, (1987)

[14] John Yen. A principled Approach to Reasoning about the Specificity of Rules. Proceedings of *AAAI-90*, Boston MA (1990)

A non-standard approach to terminological knowledge: the ITL system

Nicola Guarino
guarino@ladseb.pd.cnr.it

I would like to briefly present in this paper an approach wich deviates from the mainstream of current KL-ONE-like systems, but which still takes the original motivations of KL-ONE as the main source of inspiration. This approach has evolved in 5 years through two implemented systems, DRL [Guarino 88, 89] and ITL [Guarino 91a], and it has now gained enough maturity to show its substantial differences with respect to "standard" terminological languages. I will present here these differences in an extremely concise way, trying to relate them to the discussions made at the workshop.

A commitment to terminologies

One of the main characteristics of ITL is the fact that the objects of interest are not generic descriptions, but defined *terms*. Terminological knowledge, in our opinion, is mainly knowledge *about* terms, intended as lexical items. The knowledge about a term is expressed as a set of taxonomic relations with other terms, which are interpreted assertionally. This means that it is not possible to speak of, say, black telephones whithout having introduced the term *black-telephone:*

a telephone X is a black-telephone if color of X := [black].
color of any black-telephone := [black].

The reason of this choice is the desire to "broken" a concept description into its basic constituents, i.e. its necessary and/or sufficient conditions regarding roles. Of course, the resulting expressive power is not lower than that of standard languages, since any generic description may be given an ad-hoc concept name; the difference is that, given a certain terminology, the number of concepts which may be formed (and therefore appear within queries) is much more restricted.

Definitions vs. descriptions.

The result of the previous choice is a *fine granularity*, which allows for a great flexibility for expressing incomplete or redundant information about a given term. Taking an example reported in [Woods 90], the knowledge about the term *triangle* may be expressed by the following set of statements:

any triangle is a polygon.
(number of side) of any triangle := [3].
(number of angle) of any triangle := [3].
a polygon X is a triangle if number of side of X := [3].
a polygon X is a triangle if number of angle of X := [3].

These statements represent a (potentially incomplete) *description* of a triangle, not a *definition*. Current terminological languages may represent partial descriptions by adding so-called "rules" to definitions in order to express only-necessary conditions, but, as observed in [Doyle&Patil 91], they are not able to deal with alternate sufficient conditions. Of course, this kind of problems are important for those applications where it is necessary to *describe* a relatively stable domain (i.e., to represent its *a priori* organization), while they are less important for those applications where it is necessary to "organize a large set of objects that can naturally be represented in terms of *features* or *roles*" [Brachman&al. 90]. In my opinion, it is not just by chance that the major applications reported by the CLASSIC and LOOM groups belong to the latter category: surprisingly enough, the task of *capturing* the meaning of an object like a lexical item, which was one of the main targets of KL-ONE and KRYPTON, turns to be hard for terminological logics.

Attribute-concepts as "vivid" entities

If roles contribute to the meaning of a concept by *conditions* which they have to satisfy, it is not necessary to introduce terms like *(all R C), (some R), (atleast N R),* whose only purpose is to contribute to the meaning of a definition in a compositional way. In my opinion, they denote *artificial concepts,* lacking a "vivid" relationships with objects of interest in the domain. For instance, I cannot see how something like *(all child doctor)* can denote an object so relevant to deserve a specific construct in the language; I even doubt whether it can be called a *concept* in a cognitive sense. As we know, the reason of the introduction of these terms is merely technical: they allow us to express concepts in a nice compositional way. But if we turn to conditions, their necessity disappear and some more vivid entity is necessary: attribute-concepts. For instance, the contribution of *(all child doctor)* to the meaning of the concept *parent_of_doctors = (and person (all child doctor))* splits into two separate conditions:

any child of a parent_of_doctors has to be a doctor.
if any child of a given person is a doctor, then this person is a parent_of_doctors.

The conceptual entities which appear in the two conditions are *child of a parent_of_doctors* and *child of a given person:* both seems to be entities relevant enough to deserve a term. In ITL, the former is already a term, while the latter corresponds to the non-ground term *child of a person X*. What is interesting is that these terms denote *concepts:* in this way a necessary condition expressing a value restriction for a role can be represented homogenously to an explicit subsumption between concepts. There is therefore no need for a proliferation of concept-forming constructs.

Individuals vs. concepts

Bill Swartout raised at the workshop the issue of those concepts which, in certain cases, may be also seen as individuals. A good example may be *teacher,* which may be seen as a subconcept of *person* as well as an instance of *job*. Under this respect, ITL is very similar to OMEGA in the fact that there is no a-priori distinction between the two kinds of entities: the relevant distinction is between different ways to *refer to* a given obiect. Opaque references pertain to objects seen in a collective way, i.e. "individuals", while transparent references pertain to objects seen in a distributive way, i.e. concepts or classes. The presence or absence of *determiners* represent the syntactical tool used to implement this distinction. The result is a language where the object level and the meta level are "amalgamated".

It was argued at the workshop that this approach may be too expensive with respect to the real needs of applications, and that some ad-hoc solution may be desirable. I present here three arguments in favour of the introduction of opaque/trasparent references in ITL.

First, number restrictions can be expressed as properties of objects, without any need for ad-hoc constructs:

number of pope := [1].
number of child of bob := [3].
a person X is a parent if the number of child of X is a [1..].

Second, opaque references may be useful to solve some classical puzzles involving intensionality, like McCarthy's example of Mike's telephone number. We show in [Guarino 91a] that the practical need for intensionality is not limited to sophisticated linguistic applications, but plays a fundamental role for the representation of common-sense knowledge regarding change, causation and functional descriptions. Consider for instance the following statements:

replaced(keyboard of the macintosh of john).
replaced((keyboard of the macintosh of john):_).
increased(temperature of liquid#3).
causes(decrease of quantity of oil, increase of temperature of engine).

Finally, the distinction between opaque and trasparent references gives us the possibility to implement some form of *computational reflection,* in the sense of [Maes&Nardi 88].

Ontological adequacy

Some time was spent at the workshop on the issue of ontology. Most of us agreed on the crucial role of ontology for building large, reusable (and therefore *valuable*) knowledge bases. What is an open issue is the impact of the ontological choices on the particular knowledge representation formalism used. In my opinion, we can define three levels of increasing ontological commitment, which are briefly described below. In [Guarino 91a], I argue that only a language which satisfies the conditions associated to the three levels can be defined as *ontologically adequate.*

1. Ontological discipline. At this level, the language is neutral with respect to the ontological choices, but some guidelines are given to the user in order to build well-founded knowledge bases. These guidelines should address basic distinctions such as concepts vs. roles, concepts vs. individuals, and terminological vs. non-terminological knowledge, while proposing some naming conventions as well. Some high level, "disciplined" ontologies for various domains may be offered as an example to the users community. In [Guarino 89] and [Guarino 91b] I discuss some of the above mentioned distinctions.

2. Constrained semantics. Formal semantics of current knowledge representation languages usually accounts for a set of models which is *much larger* than the models we are interested in, i.e. real world models. As a consequence, the possibility to state something which is reasonable for the system but not reasonable in the real world is very high. In [Guarino 91a] I propose a semantics which is *not neutral with respect to some basic ontological assumptions.* Examples of these assumptions are the Attribute Consistency Postulate *any X of Y is a X* and the avoidance of an a-priori distinction between concepts and individuals.

3. Fine granularity. An ontologically adequate language should be able to express knowledge about the ontological nature of the link existing between an object and its attributes. This is especially important for what I call non-relational attributes, *3.* Fine granularity. An ontologically adequate language should be able to express knowledge about the ontological nature of the link existing between an object and its attributes. This is especially important for what I call non-relational attributes, which mainly denote *parts* or *possessions,* since their name cannot be uniquely related to the nature of the ontological relationship involved: a book, for instance, may be a member of a collection as well as a possession of a person. A possibility to speak of an object/attribute link may be its reification, as in Meta-SB-ONE [Kobsa 91]; another possibility is represented by ITL attribute-concepts.

Bibliography

Brachman, R. J., McGuinness, D. L., Patel-Schneider, P. F., Resnick, L. A. 1990. Living with CLASSIC: When and How to Use a KL-ONE-like Language. To appear in [Sowa 91].

Doyle, J., and Patil, R. S. 1991. Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence* 48.

Guarino, N. 1988. DRL: terminologic and relational knowledge in Prolog. In Y. Kodratoff (ed.), *Proc. of 8th European Conference on Artificial Intelligence (ECAI-88),* Muenchen, August 1-5, 1988. Pitman.

Guarino, N. 1989. Nature and structure of terminological knowledge: the DRL approach. *Proc. of the 1st Conf. of the Italian Association for Artificial Intelligence (AI*IA),* Trento.

Guarino, N. 1991a. A Concise Presentation of ITL. To appear on *ACM SIGART Bulletin,* special issue on Implemented Knowledge Representation and Reasoning Systems, summer 1991. An extended and revised version will appear on the *Proc. of Int. Workshop on Processing Declarative Knowledge,* Symbolic Computation Series, Springer-Verlag.

Guarino, N. 1991b. Concepts, Attributes, and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases. Italian National Research Council, LADSEB-CNR Int. Rep. 01/91.

Kobsa, A. 1991. Utilizing knowledge: The components of the SB-ONE knowledge representation workbench. To appear in [Sowa 91].

Maes, P., and Nardi, D. (eds.) 1988. *Meta-Level Architectures and Reflection.* North Holland 1988.

Sowa, J. (ed.) 1991. *Principles of Semantic Networks: Exploration in the Representation of Knowledge.* Morgan Kaufmann 1991.

Woods, W. 1990. Understanding Subsumption and Taxonomy: A Framework for Progress. To appear in [Sowa 91].

A Probabilistic Extension for Terminological Logics *

Jochen Heinsohn
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
W–6600 Saarbrücken 11, Germany
e-mail: heinsohn@dfki.uni-sb.de

Research in knowledge representation led to the development of terminological logics [18], which mainly originated from Brachman's KL-ONE [4]. In such languages, the terminological formalism (*TBox*) is used to represent a hierarchy of terms (*concepts*) which are partially ordered by a subsumption relation: If concept B is *subsumed by* concept A then the set of B's real world objects is necessarily a subset of A's world objects. In this sense, the semantics of such languages can be based on set theory. Two-place relations (*roles*) are used to describe concepts. In the case of *defined* concepts, restrictions on roles represent both necessary and sufficient conditions. For *primitive* concepts only necessary conditions are specified. The algorithm called *classifier* inserts new generic concepts at the most specific place in the terminological hierarchy according to the subsumption relation. Work on terminological languages further led to *hybrid* representation systems. Systems like BACK, CLASSIC, LOOM, KANDOR, KL-TWO, KRYPTON, MESON, SB-ONE, and YAK (for an overview and analyses see [14, 21]) make use of a separation of terminological and assertional knowledge. The assertional formalism (*ABox*) is used to represent assertions about the real world. The mechanism to find the most specific generic concept an object is an instance of and to maintain consistency between ABox and TBox is called the *realizer*.

Since, on one hand, the idea of terminological representation is essentially based on the possibility of *defining* concepts (or specifying at least necessary conditions), the classifier can be employed to draw correct inferences. On the other hand, characterizing domain concepts only by definitions can lead to problems, especially in domains where certain important properties cannot be used as part of a concept definition. As argued by Brachman [2] this may be the case in "natural" environments (in contrast to "technical/mathematical" environments). The source of the problem is the fact that in natural environments, besides their definition terms can only be characterized as having further *typical* properties or properties which are, for instance, *usually* true. If typical properties are (mis-)used to formulate definitions, this can lead to problems concerning *multiple inheritance*.[1] However, in the real world such properties often are only *tendencies*, i.e

*This work has been carried out in the WIP project which is supported by the German Ministry for Research and Technology BMFT under contract ITW 8901 8.

[1]One example commonly used to highlight these problems is known as the "quaker example": quaker

republicans "usually" are non-pacifist, for example. Tendencies as well as differences in these tendencies cannot be considered in the framework of term definitions. Several attempts have been made to cope with these observations.

Considering "typical" properties led to nonmonotonic inheritance networks, and may be viewed as "cancellation of inheritance links" or "assume to be true unless told otherwise" [26, 2, 6, 7, 17, 23]. These approaches work well if exceptions are explicitly known. However, in the case of conflicts the results can be unsatisfactory (i.e., the "multiple extension problem", compare e.g. [20]).

A solution concerning "usually true" properties is proposed by Shastri [25]. He offers a language to represent empirical information about properties of hierarchically ordered concepts. This empirical knowledge is used instead of definitional roles. His system works well in the case of exceptions and also for ambiguities. However, the system is built for handling a large amount of statistical data and is not constituted to consider terminological and statistical incompleteness. Other related work can be found in [19, 1, 15, 16, 22].

In all these proposals an algorithm comparable to the classifier for maintaining the consistency of the terminology and for reorganizing it according to implicitly existing subsumption relationships does not exist because concepts cannot be defined by necessary and sufficient conditions. The importance of providing an integration of both term classification and uncertainty representation was recently emphasized in [11, 27].[2] Yen and Bonissone [27] consider this integration from a general point of view which, for instance, does not require a concrete uncertainty model (e.g., probabilistic, fuzzy, Dempster-Shafer [12, 13]), while in [11] specific properties of an integration are demonstrated based on a concrete probabilistic model.

We propose an extension of terminological logics which allows to handle the problems discussed above [9, 10]. The extension maintains the original performance of drawing inferences on a hierarchy of terminological definitions. It enlarges the range of applicability to real world domains determined not only by definitional but also by uncertain knowledge. First, we briefly introduce $\mathcal{ALC}$ [24], a propositionally complete terminological language containing the logical connectives conjunction, disjunction, negation, as well as role quantification. By keeping the TBox semantics, which is based on term descriptions, we are able to use the classifier for extending and reorganizing the terminology. We extend $\mathcal{ALC}$ by defining syntax and semantics of *probabilistic implication* (p-implication), a construct which is aimed at considering non-terminological knowledge sources and is based on a statistical interpretation. In particular, given two concepts C_1 and C_2, the interpretation of a p-implication $C_1 \xrightarrow{p} C_2$ is given by the relative cardinality $p \overset{\text{def}}{=} : \mathcal{E}[\![C_1 \sqcap C_2]\!] : / : \mathcal{E}[\![C_1]\!] :$ where $\mathcal{E}$ maps every concept description to a subset of $2^{\mathcal{D}}$ and every role to a subset of $2^{\mathcal{D}\times\mathcal{D}}$, with $\sqcap$ denoting concept conjunction and $\mathcal{D}$ being the domain of discourse.

As demonstrated, on the basis of the terminological and probabilistic knowledge certain *consistency requirements* have to be met. Moreover, these requirements allow to infer implicitly existent probabilistic relationships and their quantitative computation [11, 5]

are pacifist, republicans are non-pacifist, and Dick is known to be both quaker and republican. The attempt to answer the question about Dick's pacifism results in the detection of a *contradiction*.

[2]Brachman [3] considers "probability and statistics" as one of the "potential highlights" in knowledge representation.

(see [1] for a logical formalism dealing with qualitative statistical information). By explicitly introducing restrictions for the ranges derived by instantiating the consistency requirements, also *exceptions* can be handled. In the categorical cases this corresponds to overriding of properties in nonmonotonic inheritance networks.

Consequently, our probabilistic extension of terminological logics takes into account uncertain knowledge arising when certain properties are e.g. usually true but not definitional. Probabilistic implication opens the way to an integration of strictly definitional knowledge and the possibility to model exceptions, which do no longer appear as contradictions [2], but as a set of weaker inequalities that guarantees the consistency of probability assignments. By separating terminological and probabilistic knowledge, processes maintaining the consistency of the terminological part remain operational. In fact, probabilistic consistency heavily depends on correct terminological subsumptions as established by the classifier.

Current investigations [9] are related to the further refinement of the rules for testing consistency and to the consideration of assertional (ABox) knowledge. The second aspect however has as consequence that two different semantics of probabilities have to be integrated, i.e., we have to cope with both universal (statistical) statements involving probabilities over domains and assertions describing particular degrees of belief by means of probabilities over possible worlds [8]. Furthermore, the way assertions about the real world are taken into account becomes different from classical hybrid representation systems: even if an instance is known to belong to a concept "with certainty", its belonging to other concepts may become uncertain. So, our framework of terminological and probabilistic knowledge requires an extension of the "classical" realizer.

References

[1] F. Bacchus. Lp, a logic for representing and reasoning with statistical knowledge. *Computational Intelligence*, 6:209–231, 1990.

[2] R. J. Brachman. 'I lied about the trees' or, defaults and definitions in knowledge representation. *The AI Magazine*, 6(3):80–93, 1985.

[3] R. J. Brachman. The future of knowledge representation. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 1082–1092, Boston, Ma., 1990.

[4] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[5] D. Dubois and H. Prade. On fuzzy syllogisms. *Computational Intelligence*, 4(2):171–179, May 1988.

[6] D. Etherington. Formalizing nonmonotonic reasoning systems. *Artificial Intelligence*, 31(1):41–85, 1987.

[7] C. Froidevaux and D. Kayser. Inheritance in semantic networks and default logic. In P. Smets, E. Mamdani, D. Dubois, and H. Prade, editors, *Non-Standard Logics for Automated Reasoning*, pages 179–212. Academic Press, New York, N.Y., 1988.

[8] J. Y. Halpern. An analysis of first-order logics of probability. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1375–1381, Detroit, Mich., 1989.

[9] J. Heinsohn. A hybrid approach for modeling uncertainty in terminological logics. DFKI Report, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany, 1991. In preparation.

[10] J. Heinsohn. A probabilistic extension for term subsumption languages. In *Proceedings of the 1st European Conference on Symbolic and Quantitative Approaches for Uncertainty (ECSQAU-91)*, Marseilles, France, October 15–17 1991. To be published by Springer-Verlag.

[11] J. Heinsohn and B. Owsnicki-Klewe. Probabilistic inheritance and reasoning in hybrid knowledge representation systems. In W. Hoeppner, editor, *Proceedings of the 12th German Workshop on Artificial Intelligence (GWAI-88)*, pages 51–60. Springer, Berlin, Germany, 1988.

[12] J. Heinsohn and J. van Loon. Numerical measures for handling uncertainty - looked at from a Bayesian perspective. Report MS-H 4795/88, Philips Research Laboratories Eindhoven/Hamburg, 1988.

[13] R. Kruse, E. Schwecke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge Based Systems: Numerical Methods.* Series Symbolic Computation -- Artificial Intelligence. Springer, Berlin, Germany, 1991. To appear.

[14] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Computer Science.* Springer, Berlin, Germany, 1990.

[15] E. Neufeld. Defaults and probabilities; extensions and coherence. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 312–323, Toronto, Ont., May 1989.

[16] G. Paass. Probabilistic logic. In P. Smets, E. Mamdani, D. Dubois, and H. Prade, editors, *Non-Standard Logics for Automated Reasoning*, pages 213–251, New York, N.Y., 1988. Academic Press.

[17] L. Padgham. *Non-Monotonic Inheritance for an Object-Oriented Knowledge-Base.* Ph.D. Dissertation No. 213, Linköping University, Sweden, 1989.

[18] P. F. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. MacGregor, W. S. Mark, D. McGuinness, B. Nebel, A. Schmiedel, and J. Yen. Term subsumption languages in knowledge representation. *The AI Magazine*, 11(2):16–23, 1990.

[19] J. Pearl. Probabilistic semantics for nonmonotonic reasoning: A survey. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 505–516, Toronto, Ont., May 1989.

[20] D. Poole. What the lottery paradox tells us about default reasoning. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 333–340, Toronto, Ont., May 1989.

[21] H.-J. Profitlich, J. Heinsohn, D. Kudenko, and B. Nebel. A comparative analysis of terminological representation systems. In *Working Notes AAAI Spring Symposium on Implemented Knowledge Representation and Reasoning Systems*, Stanford University, USA, March 26–28 1991.

[22] A. Saffiotti. A hybrid framework for representing uncertain knowledge. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 653–658, Boston, Ma., 1990.

[23] E. Sandewall. Nonmonotonic inference rules for multiple inheritance with exceptions. *Proc. of the IEEE*, 74(10):1345–1353, 1986.

[24] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48, 1991.

[25] L. Shastri. *Semantic Networks: An Evidential Formalization and its Connectionist Realization.* Pitman, London, England, 1988.

[26] D. S. Touretzky, J. F. Horty, and R. H. Thomason. A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 476–482, Milan, Italy, Aug. 1987.

[27] J. Yen and P. Bonissone. Extending term subsumption systems for uncertainty management. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, Cambridge, MA, July 1990.

On Conceptual Indexing in Terminological Systems

Carsten Kindermann
Technische Universität Berlin,
Sekr. FR 5-12, Projekt KIT-BACK
Franklinstraße 28/29, D-1000 Berlin 10, Germany
Carsten.Kindermann@cs.tu-berlin.de

Work in the area of terminological logics (TL) has always been accompanied by theoretical analysis of the worst case behavior of the respective inference algorithms. Recent results, however, suggest that investigations on *average* or *normal* case behavior of TL systems are required to estimate the practicability of the terminological approach. A system's behavior is determined by a number of factors including the completeness of the designed algorithms, decisions made when actually implementing them, and the structure of the underlying data depending on the particular application. Having worked on the implementation of several versions of the BACK system, one of my current interests is how to make TL systems adaptable in order to conduct experiments in different application environments.[1]

Experiments in Implementing BACK

During the last few years we have been experimenting with different implementations for the BACK system. Especially for the ABox we have considered several implementational alternatives, such as caching vs. recomputation, assertion time vs. query time deductions, control of the inference algorithms by data or by goals (forward and backward chaining), and have combined them in different ways for the various versions. A report on these experiments is given in [2].

I felt there were two major problems in the way we conducted the experiments:

1. For experimenting with an alternative set of choices we were forced to reimplement the core of the particular component (in this case the ABox) each time.
2. The set of criteria we had developed was useful to describe the behavior of particular implementations (e.g., "ABox V3 uses a *derivative technique*, and performs inferences at *query time* in a *backward chained* manner"). The criteria, however, were not directly transferable into running code.

[1]This work was supported by the Commission of the European Communities and is part of Esprit Project AIMS (5210).

The question is how to develop from a set of desired features the possibility to adapt the system accordingly. What I am proposing here is an approach of declaring concepts to be used for system internal purposes, and thereby to tailor terminological systems for different testbeds or different applications. The semantics of the traditional (i.e., primitive and defined) concepts remains the same. The additional declarations express how a concept is to be used to support a certain part of system functionality, e.g., that a concept should be used to index objects for faster retrieval. This approach may be useful in two ways:

- If the system is sufficiently flexible, it may be evaluated for sample data patterns that differ in characteristical properties such as breadth and depth of the concept hierarchy, maximal length of role-chains, or interrelatedness of objects. Similar test settings have been used successfully in the database area, cf. [1].
- In an application environment the adaptability of the knowledge base management system may be useful to obtain a more efficient behavior. The system may be adapted by a knowledge engineer after an analysis of the requirements and data of the domain at hand. Alternatively, a monitoring program may collect data of the actual use of the system. A heuristics-based component may then automatically adapt the system by introducing declarations for the appropriate concepts.

Conceptual Indexing for Efficient Object Retrieval

In a scenario of applying terminological systems to knowledge base management one of the frequently occurring tasks is the retrieval of instances of some query description. This task can be supported by a method we call *conceptual indexing* which essentially maintains references from concepts to their instances.[2]

The indexing structure is build up by object classification (*recognition*): For each object (at least) all indexing concepts are determined that it instantiates. For each of the indexing concepts—or, as an optimization, for the most specific ones among them—explicit references to the instances are maintained.

For a query concept (or the generic part of a query in BACK's assertional query language AQL[3]) that is equivalent to an indexing concept the set of its instances is obtained by simply following these references. For other queries the set of instances of their immediate superconcepts are intersected, and the restrictions that distinguish the query from its immediate superconcepts are used to discriminate those instances belonging to the final answer.

In the BACK system the content of the ABox can be made persistent by storing it in a relational database. Query processing can be completely delegated to the relational DBMS in case the basic query (i.e. the query's generic part) is equivalent to an indexing concept. The AQL query is then resolved by compiling it into a single SQL query. In the more complex case it may be necessary to load instances into main memory in order to

[2]Cf. also [4] for a first application of conceptual indexing.

[3]The AQL is described in detail in [5].

apply the ABox reasoning procedures. The cooperation of BACK's ABox with a relational DBMS is described in [3].

In a realistic application environment, however, it is not feasible to use all concepts to index objects. First, numerous concepts are introduced for system internal purposes, and determining their instances wastes storage without being of any interest. Second, for an indexing concept the system has to guarantee that all its instances are known. The introduction of new concepts then requires to restructure greater parts of the knowledge base. Consider as an example *abstraction descriptors*, i.e., concepts that are introduced to maintain the intensional information extracted from object descriptions. The introduction of a new object or modification of an existing one is likely to introduce/modify such a concept, and thus to cause a subsequent recomputation of the indexing structure.

The problem is solved in a natural way if we provide the means to *explicitly* divide the set of concepts into indexing and non-indexing concepts. Concepts for system internal purposes may then be introduced generously, unused abstraction descriptors may be garbage collected, and all this does not influence the indexing structure at all. Furthermore, introducing the possibility to explicitly declare concepts as being indexing allows for determining the system's query processing behavior. For instance, declaring only the primitive concepts as indexing makes TL systems behave like deductive databases where instances are kept only for base classes (corresponding to primitive concepts), while instances of derived classes (corresponding to defined concepts) are determined on demand. Alternatively, enlarging the set of indexing concepts to all user defined concepts guarantees fast retrieval of the instances of these concepts.

Annotating Concepts for System Adaptation

Making the notion of indexing concepts explicit is an example for an approach to tailor terminological systems by adding to concepts declarations that determine their system internal usage.

As another example consider the set of concepts that constitute the searchspace for recognition. BACK's recognition process is partly driven by the concepts present in the TBox. Depending on what has been asserted about an object a number of concepts serve as candidates when looking for the most special concepts the object is an instance of (cf. [2]). Preselecting among all concepts by marking those that *in general* are of interest to be tested, allows one to further determine the searchspace for recognition. A maximal choice would declare every concept as a *recognition candidate.* A minimal choice would consider only primitive concepts as candidates.[4] In this case inferences are drawn derivatively at query time, and consistency checking is performed in a limited way at assertion time. A reasonable setting is to declare as recognition candidates all indexing concepts and all concepts that correspond to the left hand side of rules (e.g., implication links, defaults, etc.). In this case the indexing structure is set up properly and the rule-like knowledge is applied forwardly.

In summary, terminological systems can be made adaptable to different test and application scenarios by annotating concepts with declarations that express how the concepts

[4]The candidate concepts for recognition must cover at least all indexing concepts.

should be used for various system internal purposes. It should be clear, however, that not every aspect of a TL system's behavior may be determined this way, and that further kinds of "switches" may be applied to obtain flexibility with respect to other aspects.

References

[1] F. Bancilhon and R. Ramakrishnan. An Amateur's Introduction to Recursive Query Processing Strategies. In J. Mylopoulos and M.L. Brodie (eds.), *Readings in Artificial Intelligence and Databases*, Los Altos (Cal.): Morgan Kaufmann, pp. 376–430, 1988.

[2] C. Kindermann. Class Instances in a Terminological Framework - An Experience Report. In H. Marburger (ed.), *Proc. of GWAI-90.* Berlin: Springer, pp. 48–57, 1990.

[3] C. Kindermann and P. Randi. Object Recognition and Retrieval in the BACK System. In S.M. Deen (ed.), *CKBS '90, Proc. of the International Working Conference on Cooperating Knowledge Based Systems*, Berlin: Springer, pp. 311–325, 1991.

[4] P.F. Patel-Schneider, R.J. Brachman, and H.J. Levesque. ARGON: Knowledge representation meets information retrieval. In *Proc. of 1st CAIA*, pp. 280–286, 1984.

[5] C. Peltason, A. Schmiedel, C. Kindermann, and J. Quantz. The BACK System Revisited. KIT Report 75, Technische Universität Berlin, 1989.

AN APPROACH TO THE INTEGRATION OF TERM DESCRIPTION LANGUAGES AND CLAUSES

by
Rüdiger Klein
Institute of Artificial Intelligence
(IfKI/ZKI)
Kurstraße 33
O–1086 Berlin

Abstract

Term description languages (TDL) in the tadition of KL–ONE provide powerful, well–formalized representational means for such KR aspects as object classes, their instances, relations and attributes, taxonomic hierarchies, etc. But normally other aspects of knowledge have to be represented, too: various logical connections and constraints, algebraic expressions, etc. Meeting all these requirements within a formal, logic–based language seems to be impossible (due to the encountered complexity). As a way out, a hybrid reasoning approach will besuggested. This implies the specification of some well–defined restriction in the concept representation capabilities: relational terms are excluded from concept definitions, instead relations will be defined including concept assignments to their arguments as a kind of consistency conditions. Though this results in reduced concept description capabilities, the overall expressiveness will be improved by the increased representational means on the side of relations and logical expressions.

In this way, in many cases an adequate representation of the well–structured object–level knowledge can be achieved, providing a "background theory" for more general knowledge (represented, for instance, as a set of clauses).

Configuration problems will be used as an illustration of our considerations: due to the well–structured, exact, and (relatively) complete object level knowledge they provide a suitable testbed. The constructive problem solving (CPS) will be outlined as an abductive approach to configuration problem solving. Taxonomic reasoning, which is an essential part of the overall configuration problem solving, can be formalized within this paradigm.

I. Introduction

A central issue in recent AI research is the developement of formal, well–defined knowledge representation schemes [Levesque 86]. Such a scheme should provide *epistemologically adequate* representational capabilities [Brachman, Schmolze 85], a *declarative* (Tarski–style) semantics, and a set of explicitly specified inference rules, which allow to address soundness, completeness and complexity issues.

Various formal, logic–based knowledge representation approaches have been investigated: order–sorted logic [Walther 87; Cohn 87], feature logic [Smolka 88; Schmidt–Schauß, Smolka 88], term–description languages (TDL) in the tradition of KL–ONE [Brachman, Schmolze 85] or approaches integrating various schemes (for instance [Pletat, Luck 89; Ait–Kaci, Podelski 91]).

All these formal schemes focus on the representation of object *classes*: their attributes, the taxonomic hierarchies they are involved, etc. *Relations* are mainly considered as representational means with respect to these classes (by the inclusion of relational terms like ∃rel.c or ∀rel.c into concept specifications). But in many fields of application relations should be a representational means in their own (a kind of "first–order citicens" in the world of representations), and both, relations as well as concept terms, should be incorporated into logical expressions, in order to represent general logical connections and constraints as an essential part of the definitional knowledge.

Recent results of theoretical investigations in these formal knowledge representation schemes revealed a basic conflict between expressiveness and inferential complexity [Nebel 90; Nebel 90a; Hollunder, Nutt 90]. Every scheme providing sufficient expressiveness may easily result in intractability (or even undecidability [Schild 88; Patel–Schneider 89]) of the inferences.

As a consequence, in recent years considerable interest has been grown in *hybrid reasoning* techniques [Baader et al. 90; Frisch, Cohn 91], integrating various knowledge representation and reasoning techniques in a *well–formalized* way.

Two main questions arise with such hybrid schemes:

- Which knowledge may be represented in which way?
- How may the various forms of knowledge interact?

The second question has been addressed in some general theoretical approaches: for instance theory resolution [Stickel 85], the substitutional frame concept [Frisch 89] as a generalization of order–sorted unification [Walther 87, Cohn 87], or constraint resolution [Bürckert 90; Baader et al. 90]. Of course, their *applicability* will be affected by the answer found to the first question.

In conjunction with further theoretical investigations of hybrid knowledge representations more *practical experiences* have to be gained in order to get *expressive and efficient* hybrid reasoning schemes. That's the main intention of this paper.

In order to retain the formal foundation of knowledge representation and to avoid unnecessary complexity, this hybrid representation should include some *well–defined restrictions* in the representational capabilities and in the inferences [Klein 91]. A scheme will be suggested here, which allows the (restricted) integration of *term descriptions* and *logical expressions* (in the form of clauses).

In chapter 2 a motivation for our approach to integrate these two essential aspects of knowledge representation will be given. Chapter 3 outlines a formal definition of the syntax and the declarative semantics of our knowledge representation, including the restrictions in the knowledge representation and inferences specified. Chapter 4 contains a description of the taxonomic reasoning based on these suggestions, followed by a discussion in the final chapter 5.

II. Motivation

Term description languages provide expressive means to object class (concept) and instance representations. But a further increase of expressiveness, for instance by the integration of clauses in the "traditional" TDL manner, easily results in intractability [Schild 89; Quantz 90].

As a way out of this conflict, we suggest a *restriction* of term description capabilities: relational expressions will not be allowed as part of concept definitions. This, of course, may not always be adequate. But in many domains, especially such "well–structured" ones like configuration, scheduling, model–based diagnosis, etc., the object classes may be defined *per se*, i.e., by their own features and attributes, without being influenced *conceptually* by reations to other objects. Relations, in contrast, should be considered here as *first–order* representational means. They will be characterized additionally by concept assignments to their arguments, providing necessary pre–conditions for objects to be consistently involved in such a relation.

Based on these representational means, which provide a kind of "background knowledge", logical expressions (for instance in the form of clauses) could be used to represent general logical connections and constraints.

We suggest the following scheme of the definitional knowledge:

- The *object–centered knowledge representation* (concept descriptions) will include Boolean expressions on other concepts, feature terms (selections), and feature (dis–)agreements [Smolka 88; Schmidt–Schauß, Smolka 88]. But in contrast to other term description languages, concept–defining terms will *not* contain *any relational expressions* (like ∃rel.c or ∀rel.c)[1].
- As part of the *definitional* knowledge, relations will be defined including concept assignments to their arguments (in analogy to order–sorted logic [Walther 87; Cohn 87]). These concept assignments provide *necessary pre–conditions* for objects to be in such a relation to each other.
- Both concept and relational expressions may be included in clauses, expressing general logical constraints.

In conjunction with this definitional knowledge, a set of assertions will be used to represent a *concrete* problem/solution. The interaction between both aspects of knowledge representation will be described in chapter 4.

1. The distinction between *features* and *relations* (roles) as taken here will not be only a formal one (functional versus relational expressions), but basically one reflecting the *different intended meaning*: a feature will uniquely be assigned to one of the objects, whereas a role specifies a relation between various objects (see for instance the example described in chapter 5.).

The overall representation scheme of the *definitional knowledge* adopted here has been outlined in fig. 1.

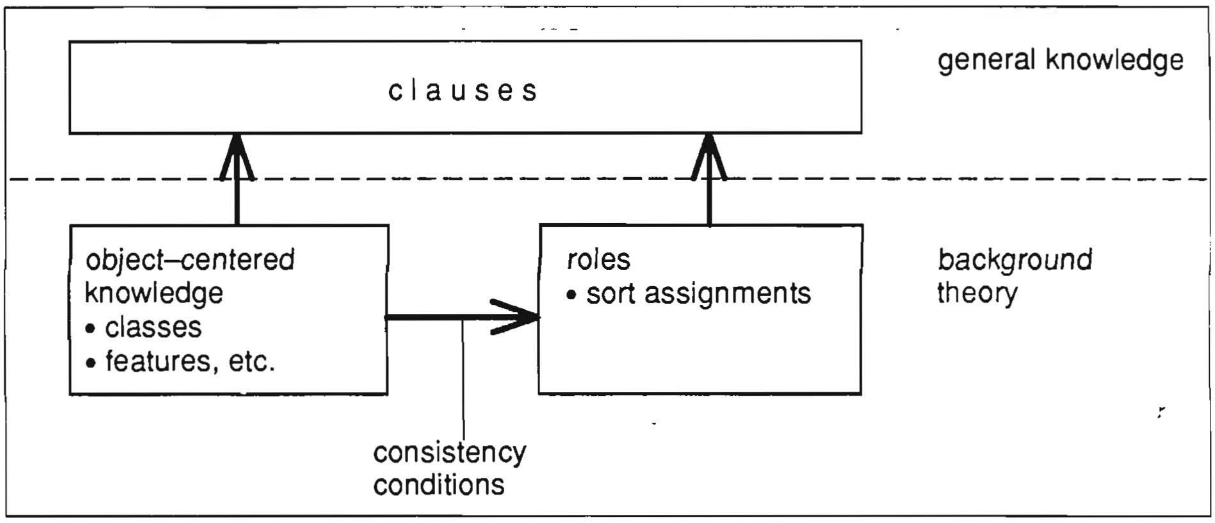

Fig. 1: An outline of the definitional knowledge representation

The restrictions in the concept description capabilities mentioned above (no relational expressions in concept descriptions) will be accompanied with a *restriction of the terminological inferences*: no relational expressions could be used in subsumption calculations. But this restriction will be compensated by an increased inferential power on the side of logical expressions, which involve concept and role expressions (as a background theory, for instance, to a modified[2] constraint resolution [Bürckert 90; Baader et al. 90]). This will be described in chapter 4.

III. Formal Definition of Object–Level Knowledge Representation

As a concrete specification of the ideas discussed in the preceding chapter, we define the following knowledge representation scheme:

- The object class descriptions will be represented by a sort[3] hierarchy SH;
- A set REL^{SH} of relations with SH sort assignments to their arguments (representing consistency conditions);
- A set $CL+^{SH}$ and a set $CL-^{SH}$ of of definite and of negative clauses[4], respectively, on sort and relational expressions, representing general logical connections.

Object Class Representations

We start with the specification of the object–class representation in a sort hierarchy SH:

Definition: sort signature

A sort signature Σ is a triple: $\Sigma = <S, F, A>$, with

- S = a set of sort symbols (concepts);

2. As will be explained in chapter 4, configuration problem solving will be described in terms of abductive reasoning, which implies some modifications of standard resolution and unification [Klein 91a].
3. The notions sort and concept will be used synonymously.
4. The restriction to definite and negative (Horn) clauses will be motivated in the next chapter.

- F = an S–family of sets of feature symbols: $F_{s,s'}$; and
- A = an S–family of sets of atoms (constants): A_s

Out of these primitive building blocks more complex *sort expressions* may be formulated:

Definition: sort expressions

Given a sort signature $\Sigma = <S, F, A>$, a set SE of sort expressions may be defined from this signature as follows:

- every sort symbol is a sort expression: $S \subseteq SE$;
- se $\wedge$ se' will be in SE;
- se $\vee$ se' will be in SE;
- $\neg$se will be in SE;
- $\{a_1, a_2, \ldots a_n\}$ will be in SE;
- f:se will be in SE; and
- p=q and p$\neq$q will be in SE

with se, se' being sort expressions, f a feature symbol, p and q being feature paths [Smolka 88], and every a_i a constant.

A further increase in expressiveness will be gained by the specification of a set SC of *sort constraints* allowing to relate different sort expressions. Two forms of sort constraints will be provided: *sort definitions* (':=') in analogy to KL–ONE's defined concepts [Brachman, Schmolze 85], and *sort restrictions* (':<'), comparable to *primitive concepts* in KL–ONE.

Finally, a *sort hierarchy* can be defined on these representational means as a representation of a taxonomic hierarchy:

Definition: sort hierarchy

A sort hierarchy SH is a pair SH = $< \Sigma$, SC>, with

- Σ being a sort signature; and
- SC being a set of sort constraints.

These representational means of object classes are basically a conjunction of feature logic constructs with term description capabilities [Pletat, Luck 89].

Relations

A set REL^{SH} of relations will be specified as part of the *definitional* knowledge. Each relation in REL^{SH} (or for short, SH–relation) will be provided with SH sort assignments of its arguments (in analogy to order–sorted logic [Walther 87]). These assignments will be interpreted as necessary pre–conditions for every object, which can be included in such a relation. Relational expressions (like $\exists$rel.c or $\forall$rel.c) will be excluded from concept definitions.

Definition: relations

Given a sort hierarchy SH, a set REL^{SH} of relations will be defined. Each SH–relation rel in REL^{SH} will be assigned a string s1,s2,..., sn of sort expressions (s1,s2,...,sn$\in$SE):

$$rel_{s1,\, s2,\, \ldots\, ,\, sn}$$

specifying the sort assignments to each of its n arguments.

Semantics

Based on these syntax specifications, a declarative Tarski–style, set–theoretic semantics may be defined as usual. We start with the definition of an *interpretation*:

Definition: Interpretation

Given a sort signature $\Sigma = <S, F, A>$, a sort hierarchy SH, and a set REL^{SH} of SH–relations, an interpretation I is a pair $I = <\Delta^I, I^\cdot>$, with Δ^I being a set (the *domain* of the interpretation), and $I^\cdot$ being an *interpretation function* having the following properties:

- every sort expression $se \in SE$ will be assigned a subset $\Delta^I{}_{se}$ of the domain Δ^I:

  $I(se) = \Delta^I{}_{se}$, with $\Delta^I{}_{se} \subseteq \Delta^I$;
- every feature $f_{s,se} \in F$ will be assigned a function $f^I{}_{s,s'}$, mapping elements of its domain interpretation $I(s)$ to elements of its range interpretation $I(s')$:

  $I(f_{s,s'}) = f^I{}_{s,s'} : I(s) \rightarrow I(s')$;
- every attribute $a_s \in A_s$ will be assigned an element of the corresponding sort interpretation $I(s)$:

  $I(a_s) = a^I{}_s$, with $a^I{}_s \in I(s)$; and
- every relation $r_{s1,s2,\ldots,sn} \in REL^{SH}$ will be assigned a subset of the Cartesian product on the corresponding sort interpretations:

  $I(r_{s1,s2,\ldots,sn}) \subseteq I(s_1) \times I(s_2) \times \ldots \times I(s_n)$.

Of course we are normally not interested in *any* interpretation, but in those fulfilling the conditions specified by sort expressions and sort constraints [Nebel 90a; Pletat, Luck 89; Schmidt–Schauß, Smolka 88]. Any such interpretation will be called a *model* of SH.

A sort hierarchy SH will be called *inconsistent*, if it containes a sort symbol, which in every model interpretation will be assigned the empty set as only possible interpretation. Inconsistent sort hierarchies will not be considered in the following.

Based on these semantical considerations, the *subsumption* relation between sort expressions can be defined (in the usual TDL manner):

Definition: subsumption

Given a sort hierarchy SH, a sort expression $se \in SE$ will be said to *subsume* (or SH–subsume) another sort expression $se' \in SE$, iff for any interpretation $I = <\Delta^I, I^\cdot>$ being a model of SH holds:

$I(se') \subseteq I(se)$

This subsumption relation between se and se' will be written

$se' \leq_{SH} se$.

This notion of subsumption also provides the *semantical foundation* to the consistency conditions of *relational expressions* and *clauses*: having for instance a relation rel *defined* in REL^{SH} with sort assignments s1,s2,...,sn to its n arguments:

$rel_{s1,s2,\ldots,sn} \in REL^{SH}$

any relational term rel(o1,o2,..,on) will only be consistent, if each of its arguments o_i (i = 1,...,n) has a sort assignment s_i' (i = 1,...,n) fulfilling the *well–sortedness conditions* [Frisch 89][5]:

$s_i' \leq_{SH} s_i \quad (\forall i = 1,...,n)$

Thus the sort hierarchy SH will be used as a *sort theory* [Frisch 89], which has to be fulfilled by every semantics. The resulting *hybrid entailment* will be written (as usual) $|=_{SH}$.

IV. Hybrid Inferences: Keeping the Solution Consistent

This hybrid knowledge representation scheme could be used in a traditonal, deductive way: in theory resolution [Stickel 85], constraint resolution [Bürckert 90], within the substitutional framework [Frisch 89], etc. But in order to demonstrate the usefulness of our scheme in full extent, a special problem solving approach will be introduced, which puts its main emphasis on the *consistent construction and manipulation of a set of assertions*. This approach greatly utilizes the various aspects of consistency, which are represented within the *definitional* knowledge: the concept descriptions, the relation specifications (including concept assignments to their argumenmts), and as sets of clauses.

Configuration problem solving has been shown to be an interesting field of applications of these ideas[Klein 91+91a], because problem solving here is mainly a *synthetic* process: a solution will be generated, which allows to fulfil the goals formulated as well as the constraints defining consistency. As a result of this consistency maintenance, a kind of taxonomic reasoning occurs. This essential element of configuration problem solving [Peltason 89; Klein 90a] is closely connected to the concept representation chosen.

As a basis of our discussion we'll outline a *formal approach* to configuration problem solving (introduced in [Klein 90]), called *constructive problem solving* (CPS):

Definition: constructive problem solving

- The *definitional knowledge* characterizing a configuration domain will be represented by a sort hierarchy SH, a set REL^{SH} of relation specifications and sets $CL+^{SH}$ and $CL-^{SH}$ of definite and negative clauses, respectively, on sort and relational expressions.
- A *concrete configuration problem* will be represented by a set $GOAL^{SH}$ of atomic goal expressions (interpreted as conjunction).
- *Solving* a configuration problem formulated in this way means to generate a solution SOL^{SH} being a set of assertions (a database of ground atomic expressions representing objects and relations between them). This set SOL^{SH} has to fulfil the following *formal conditions* in order to be a correct solution:
- $SOL^{SH} \cup CL+^{SH} |=_{SH} GOAL^{SH}$;;; the goal has to be fulfilled
- $\forall c \in CL-^{SH}$: $SOL^{SH} \cup CL+^{SH} |=_{SH} c$;;; the constraints must be fulfilled

5. This provides a semantical extension of the original, synactically defined well–sortedness of order–sorted logic [Walther 87; Cohn 87].

Up to now there is no comprehensive formalization of this basically *abductive*[6] approach. This is at least in part due to a missing general theory of abduction (on the predicate logic level) [O'Rorke 90+91; Levesque 89; Selman, Levesque 90].Of course, also some meta–criteria should be fulfilled by the abductive reasoning process. The generated solution database SOL^{SH} has to fulfil, for instance, a kind of minimality criterion with respect to the set of objects introduced and to relations specified between them.

The restriction of the clauses to definite and negative ones may significantly reduce complexity (without loosing too much in expressiveness [Kowalski 90]): the negative clauses allow to represent inconsistency explicitly, without affecting the "positive side" of the problem solving[7] (realized by the definite clauses). The set of definite clauses will be treated as a *complete decription* of the positive literals contained [Klein 90; Console et al. 90].

These and other formal issues of the CPS approach will be discussed elsewhere [Klein 91b].

The solution database SOL^{SH} will formally be defined as follows:

Definition: solution SOL^{SH}

Given a sort hierarchy SH and a set REL^{SH} of relation definitions, a solution database SOL^{SH} is a set of assertions (ground atomic expressions):

- object descriptions: a:s – with a being an object and s being a sort expression: $s \in SE$; and
- relational expressions : $rel(o_1,..., o_n)$ – with $rel_{s1,...,sn} \in REL^{SH}$ being a relation, and $o_1{:}s_1'... o_n{:}s_n'$ being objects in SOL^{SH} fulfilling the well–sortedness conditions:
  $\forall i = 1,...,n{:}\ s_i' \leq_{SH} s_i$.

The semantics can be extended in the usual way in order to capture the assertional terms, hybrid entailment, etc. [Nebel 90a].

Solution Consistency

One of the essential points with this kind of constructive problem solving is *database consistency*: having an object description

$a{:}s_1$

in the solution database SOL^{SH}, and an assertion

$r(a,b)$

with $r_{s,s'} \in REL^{SH}$ being a relation defined as part of the definitional knowledge, the object a must have a sort assignment $s_1 \wedge s$ as a necessary pre–condition of a *consistent* database SOL^{SH}:

$$\{a{:}s_1,\ r(a,b)\ ...\} \models_{SH} a{:}\ s_1 \wedge s$$

Of course, $s_1 \wedge s$ has to be consistent, too.

6. For a discussion of abductive reasoning in configuration problem solving and the relation between deductive and abductive inferences see for instance [Coyne et al. 90; Poole 90; Klein 90] and references cited there.
7. At least as long as the solution generated will be consistent.

Exactly this incremental specification of objects and of relations between them will be done by the abductive inferences in the constructive problem solving. This results in a kind of *taxonomic reasoning* by monotonicly restricting the sort expressions of the objects in order to keep the solution database consistent.

V. Discussion

The basic conflict between expressiveness and complexity of inference operations encountered in term description languages strongly stimulated the investigation of hybrid reasoning schemes [Frisch, Cohn 91].

Our approach to a hybrid, theoretically well–defined knowledge representation has mainly been based on a well–defined restriction in the expressiveness of the concept description capabilities: relations (roles) have been excluded from concept descriptions. This seems to be adequate in those cases, where due to the well–structuredness of the object level knowledge the object classes can be described *per se*, without relations to other objects [Klein 91+91a]. This well–structuredness also implies, that it will be useful to define relations in conjunction with concept assignments to their arguments (providing a kind of consistency information). As a result, the integration of this "background" definitional knowledge into more general logical expressions is possible (in order to represent the various logical connections and constraints being relevant there).

Expressive means on the side of relations have been provided in other systems, too [BACK 89; Nebel 90a and references cited there]: domain and range restrictions of roles, and various role–forming operators (like composition, inverse roles, transitive expressions). Despite the fact, that (normally) these representational means are not fully integrated into the subsumption inferences (due to the encountered computational complexity [Quantz 90]), the main problem with these approaches is the *conceptual view* applied: roles – primitive or defined – are taken at first glance as concept–describing capabilities. In our approach, more emphasis has been put onto relations as *first–order representational means.*

This provides an expressive and well–defined opportunity to integrate concept descriptions and relations into more general logical expressions.

Because relational expressions heve been excluded with purpose from concept description, it makes no sense to take them into account in subsumption inferences. As a result, these inferences will be incomplete.

On the other side, the concept descriptions in the sort hierarchy, the relation specifications including concept assignments to relation arguments, and the clauses enable a great variety of inferences on the side of assertions. This results in a shift of emphasis from definitional to assertional inferences. The main aspect of the *hybrid reasoning* would be to keep *assertions consistent.*

Of course, in certain cases the restrictions specified here would be a disadvantage. Thus the question arises: How could we get things as *expressive* as needed, and keep them *tractable* and *theoretically well-defined*? Here, as in many other cases of hybrid inference systems, the main problem seems to be the *control of inferences* [Kowalski 90]. As a result of an efficient control, the *practical* complexity may be kept tractable. Take for instance subsumption inferences: knowing the instance a to be an element of sort s (a:s), we may answer a question, whether a:s' holds, by searching for a subsumption relation between sorts s and s': $s \leq s'$. But answering this question may be done by deducing a:s' directly (only for the special case of object a), too, which could be much less expensive[8]. In our approach, we have generally excluded relational expressions from concept descriptions (without any possibility of control). Having a suitable way to *control* the inference process, this may allow to take relations (roles) into account as concept defining terms in *some special cases*, without increasing complexity *in general*.

The constructive problem solving (CPS) approach has been used in order to demonstrate the consistency-based reasoning capabilities of the hybrid knowledge representation scheme introduced here. It may be considered as a formal description of configuration problem solving. Due to the lack of a general theory of abduction (at least on the predicate-logic level), only some aspects of this approach could be demonstrated here. The discussion of other essential aspects of CPS (like variable treatment, minimum model semantics, inference rules) will be performed elsewhere [Klein 91b].

Acknowledgements

Many thanks to the collegues from the ESPRIT project KIT-BACK at TU Berlin, especially Jochen Quantz, for some very helpful discussions.

References

[Ait-Kaci, Podelski 91]
Ait-Kaci, H., and Podelski, A.: Is there a Meanng to LIFE?, 2nd International Workshop on Terminological Logic, Schloß Dagstuhl, May 1991, Statements of Interest, IBM Report, IWBS Stuttgart, 1991.
[BACK 89]
Luck, K.v. et al.: "The BACK System Revisited", KIT-Report 75, TU Berlin, 1989
[Baader et al. 90]
Baader, F., Bürckert, H.-J., Hollunder, B., Nutt, W., and Siekmann, J.H.: Concept Logic, in: [Lloyd 90], pp. 177-201.
[Bürckert 90]
Bürckert, H.-J.: A Resolution Principle for Clauses with Constraints, in: M. Stickel (ed.): Proc. 10th Conf. on Autom Deduction, Kaiserslautern, 1990.
[Brachman, Schmolze 85]
R.J. Brachman, J.G. Schmolze: An Overview of the KL-ONE Knowledge Representation System, Cognitive Science 9 (85) 171-216.
[Cohn 87]
A.G. Cohn: "A More Expressive Formulation of Many-Sorted Logic", J. of Autom. Reasoning, 3/2 (87) 113
[Console et al. 90]
Console, L., et al.: A Completion Semantics for Object-Level Abduction, in: [O'Rorke 90], pp. 72-76.
[Coyne et al. 90]

8. though the opposite case may be true as well

Coyne, R., et al.: Knowledge–Based Design Systems, Addison Wesley, Reading (Mass.), 1990._

[Frisch 89]

Frisch, A.: A General Framework of Sorted Deduction, in: Brachman, R., Levesque, H., and Reiter, R. (eds.): Proc. of the First International Conference on Principles of Knowledge Representation, Toronto, May 1989, pp. 126–136, Morgan Kaufman Publ., 1989.

[Frisch, Cohn 91]

Frisch, A., and Cohn, A.: Thoughts and Afterthoughts on the 1988 Workshop on Hybrid Reasoning, AI Mag. (Special Issue), Jan. 1991, pp.77–87.

[Hollunder, Nutt 90]

Hollunder, B. and Nutt, W.: Subsumption Algorithms for Concept Languages, Report 90–04, DFKI.

[Klein 90]

Klein, R. : Problem solving as database construction, Proc. 4. Workshop "Planen und Konfigurieren", FAW Bericht, Ulm, April 1990.

[Klein 90a]

Klein, R.: Towards an Integration of Knowledge Based Systems with Computer–Aided Design, in: U. Geske, D. Koch (eds.): Contributions to AI, Akademie–Verlag, Berlin, 1990.

[Klein 91]

Klein, R.: Model Represntation and Taxonomic Reasoning in Configuration Problem Solving, German Workshop on AI, to appear in Springer Lecture Notes in AI, Springer, Berlin, 1991.

[Klein 91a]

Klein, R.: Towards a Logic–Based Model Representation in Configuration Problems, ÖGAI91 Workshop on Model Based Reasoning, Wien, Sept. 91

[Klein 91b]

Klein, R.: Constructive Problem Solving, subm. to the 8th Deduction Workshop, Berlin, Oct. 1991.

[Kowalski 90]

Kowalski, R.: Problems and Promisses of Computational Logic, in: [Lloyd 90], pp. 1–36

[Levesque 86]

Levesque, H.: Making Believers out of Computers, AI30/1(1986)81–108.

[Levesque 89]

Levesque, H.: A knowledge–level account of abduction, Proc. IJCAI–89, pp.1061–1066, Detroit, 1989

[Lloyd 90]

Lloyd, J.W.: Computational Logic, Proc. of the ESPRIT Basic Reasaerch Activities Symposium, Bruxels, Nov. 1990, Springer, Berlin, 1990.

[Nebel 90]

Nebel, B.: Terminological Reasoning is Inherently Intractable, AI Journal 43/2(1990)235–250.

[Nebel 90a]

Nebel, B.: Reasoning and Revision in Hybrid Representation Systems, Lecture Notes in AI 422, Springer, Berlin, 1990.

[O'Rorke 90]

O'Rorke, P.: Automated Abduction, Working Notes, 1990 AAAI Spring Symposium, Stanford–Univ., TR–90–32

[O'Rorke 91]

O'Rorke, P.: Review of AAAI–90 Spring Symposium on Automated Abduction, SIGART Bulletin 1/3(1991), pp.12–17.

[Patel–Schneider 89]

Patel–Schneider, P.: Undecidability of Subsumption in NIKL, AI 39(1989)263–272.

[Peltason 89]

Peltason, C.: "Wissensrepräsentation für Entwurfssysteme", Diss. TU Berlin, 1989.

[Pletat, Luck 89]

Pletat, C. und v. Luck, K.: Die Wissensrepresentationssprache SORT–LILOG, IWBS–Report 89, IBM Stuttgart, 1989

[Poole 90]

Poole, D.: Hypo–Deductive Reasoning for Abduction, Default Reasoning and Design, in: [O'Rorke 90], pp. 106–110.

[Quantz 90]

Quantz, J.: Modeling and Reasoning with Defined Roles in BACK, KIT–BACK Report 84, TU Berlin, 1990.

[Schild 88]

Schild, K.: Undecidability of Subsumption in U, KIT–Report 67, TU Berlin, Oct. 88

[Schild 89]

Schild, K.: Towards a Theory of Frames and Rules, KIT–Report 76, TU Berlin, Dec. 89

[Schmidt–Schauss, Smolka 88]

Schmidt–Schauss, M. and Smolka, G.: Attributive Concept Description with Unions and Complements,SEKI Report 88–21, Universitaet Kaiserslautern, Dec. 88

[Searls, Norton 90]

Searls, D.B. and Norton, L.M.: Logic–Based Configuration with a Semantic Network, Journal of Logic Progr. 8(1990)53–73.

[Selman, Levesque 90]

Selman, B., and Levesque, H.: Abductive and Default Reasoning: A Computational Core, Proc. AAAI–90 , pp.343–348.

[Smolka 88]

Smolka, G.: A Feature Logic with Subsorts, IBM Report 33, IWBS Stuttgart, May 1988.

[Stickel 85]

Stickel, M.: Automated Deduction by Theory Resolution, J. Autom. Reas, 1(85) 333

[Walther 87]

Walther, C.: A Many–Sorted Calculus with Resolution and Paramodulation, Morgan–Kaufman Publ., 1987.

Reification in SB-ONE

Alfred Kobsa
Dept. of Computer Science
University of Saarbrücken
D-6600 Saarbrücken 11
GERMANY
kobsa@cs.uni-sb.de

Terminological representation systems are neutral with respect to what kinds of things in the domain to be modeled should be regarded as individuals (and hence be represented by concepts), and what should be regarded as relations (and thus be represented by roles). The decision as to how to represent certain conceptual knowledge is entirely left to the knowledge engineer who models a domain.

It is true that [1] present a guideline for the discrimination between concepts and roles, namely to determine "as to whether a description can stand on its own without implying an unmentioned object related to the object in question". If so, the description at hand would constitute a concept, otherwise a role of the unmentioned object. However, this guideline seems to be frequently violated by current representational practice in the field of natural-language processing, where a number of conventions pertaining to the concept/role dichtonomy have emerged since certain forms of representation turned out to be advantageous for natural-language applications. For instance, if actions are regarded as individuals rather than relations, it is easier to specify the relationship between the semantic cases [2] of the natural-language verbs which describe these actions and the attributes of these actions (such as the agent, object, etc.). The example of action representation seems to contradict the above guideline of [1], since an action like 'give' is related to unmentioned objects (e.g., the object being given, as well as the agent of this action), but is nevertheless represented by a concept. Another example of conflicting guidelines in the decision between representation with concepts or representation with roles are "societal persons": while [8][238] regard 'father' and 'mother' as "roles persons play in the concept 'family' ", these notions are favorite examples of concepts in the KL-ONE literature. Things become even worse in other fields of AI (such as expert systems) or in the area of conceptual modeling for databases, where no convention at all seems to exist as yet with respect to what to regard as a frame or schema, and what as a slot or schema attribute.

This arbitrariness in the representation of knowledge via concepts or roles does not seem harmful as long as knowledge bases are used in isolation only. As soon as one wants to combine the conceptual knowledge of two or more knowledge bases, representational variants for the same knowledge pose serious problems. One solution would be to decide on one variant, and suppress the other in the combined knowledge base. This, however, means that processes which expect the suppressed version can no longer operate on the combined knowledge base.

The solution pursued in the development of SB-ONE [3, 6, 5, 7] was to enhance the language in such a way that objects in the world can be regarded both as individuals and as pairs of a relation, so that *the same knowledge can be represented both through concepts and through roles.* For achieving this, both the interpretative domain and the representational elements of SB-ONE must be augmented (the resulting language was coined 'Meta-SB-ONE'). Different linearly ordered *ontological levels* are introduced into the interpretative domain $\mathcal{D}$, and each individual in $\mathcal{D}$ is assigned to one of these levels. A *reification relation* is introduced between higher-level individuals and pairs whose elements belong to a lower level (which expresses that a higher-level individual "stands for" a lower-level pair). Two new representational elements are introduced on the general level, namely so-called *metaconcepts* (they possess two special roles). and the *reif* relation between metaconcepts and roles. If a role is *reified* using the *reif* relation. the assertions expressed by this role become additionally represented by the metaconcept and its special roles. An example is given in Fig. 1, which also illustrates the graphical notation of some SB-ONE and Meta-SB-ONE knowledge representation elements. In this example, the role 'has-child' of PERSON with value restriction *CHILD* has been reified into the metaconcept 'PARENTSHIP' with two special roles whose value restrictions are PERSON and *CHILD*. respectively.

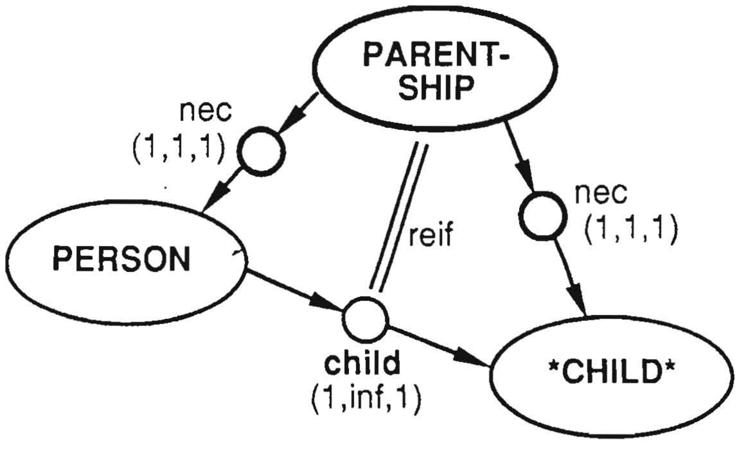

Figure 1: Example of a metaconcept and the *reif* relation

Metaconcepts and the *reif* relation are governed by a number of syntactic constraints, the most important being that one special role of the metaconcept of a role r must have r's domain concept as its value restriction, and the other r's value restriction as its own value restriction. The denotation function Δ, which maps SB-ONE concepts into the interpretative domain $\mathcal{D}$ must be slightly redefined in that all individuals of $\mathcal{D}$ which are in the denotation of a concept must pertain to the same ontological level. Thus each concept maps into a subset of the individuals of a single ontological level of $\mathcal{D}$. In a well-formed Meta-SB-ONE knowledge base. a separate root concept gc_{0_i} is introduced for each ontological level i. All these root concepts are disjoint from each other. An additional requirement for roles of a concept of a certain level is that only concepts of the same or a lower level may be employed as value restrictions. This constraint, together with the different root concepts for each ontological level, guarantee that the ontological distinctions of the interpretative domain are also syntactically observed (e.g.

by the classifier). Reification may occur arbitrarily often, i.e. roles of metaconcepts may again be reified, etc. A set of individuals of the next higher ontological level in the interpretative domain is thereby described each time. It is doubtful, however, whether a double or even multiple reification makes sense. In practical applications, single reification will most probably be sufficient.

References

[1] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. Alperin Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks: Exploration in the Representation of Knowledge*. Morgan Kaufmann, San Mateo, CA, 1991.

[2] C. J. Fillmore. The case for the case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart and Winston, New York, 1968.

[3] A. Kobsa. The SB-ONE knowledge representation workbench (extended version). Memo 50. SFB 314: AI - Knowledge-Based Systems, Dept. of Computer Science, Univ. of Saarbrücken. Saarbrücken, Germany, 1990.

[4] A. Kobsa. First experiences with the SB-ONE knowledge representation workbench in natural-language applications. *SIGART Newsletter*, Summer, 1991.

[5] A. Kobsa. Reification in meta-SB-ONE: Bridging the object/relation dichotomy. Unpublished Manuscript, Dept. of Computer Science, Univ. of Saarbrücken, Saarbrücken, Germany., 1991.

[6] A. Kobsa. Utilizing knowledge: The components of the SB-ONE knowledge representation workbench. In J. Sowa, editor, *Principles of Semantic Networks: Exploration in the Representation of Knowledge*. Morgan Kaufmann, San Mateo, CA, 1991.

[7] A. Kobsa. The SB-ONE knowledge representation workbench. *Computational Intelligence*. (submitted), 1992.

[8] H. Trost and I. Steinacker. The role of roles: Some aspects of real world knowledge representation. In *Proc. of the 7th IJCAI*, pages 237–239, Vancouver, Canada, 1981.

Statement of Interest, Dagstuhl Workshop

Robert MacGregor
USC/Information Sciences Institute
Marina del Rey, California, United States
macgreg@isi.edu

Below, I list a few topics that are of particular interest to me, and which I would like to be the subject of discussion during the workshop.

1 Semantic Choice Points

The recently circulated KRSS specification emanating from a US-based knowledge representation standards effort, and a proposal for a terminological logic emanating from a group at DFKI, Kaiserslautern represent initial attempts to bring some kind of order to the growing field of terminologically-based systems. I am hoping that, among other things, these specifications will inspire discussion of a number of semantic issues that previously have been ignored for the most part by the community, probably because too few systems had reached a point where they had to bite the bullet and make a particular semantic choice. Below, I list some of these exemplary issues. In each case, we have made an explicit design decision in LOOM (i.e., we have lots of bullet fragments lying around).

1. **Temporal semantics**. Any system that supports either role closure or retraction implicitly defines some sort of temporal semantics. This semantics ought to be made explicit. The need for an explicit semantics becomes even more evident when behavioral constructs such as production rules are introduced into the representational framework.

2. **Retraction semantics**. A number of systems implement retraction of facts. From discussions I've had with Peter Patel-Schneider and Ron Brachman, it appears that LOOM and CLASSIC have adopted a very similar semantics. It would be valuable to survey each of the systems represented at the Workshop to find out if there is in fact a consensus on retraction semantics, or if significant differences of opinion exist.

3. **Skolem individuals**. The generation of objects representing skolem individuals might be considered simply as an implementation detail not deserving of attention by a knowledge level language specification. However, we have users that have explicitly requested that LOOM generate skolem objects in certain situations. Hence, I would like to explore the question of when the system should support retrieval of skolem individuals.

by the classifier). Reification may occur arbitrarily often, i.e. roles of metaconcepts may again be reified, etc. A set of individuals of the next higher ontological level in the interpretative domain is thereby described each time. It is doubtful, however, whether a double or even multiple reification makes sense. In practical applications, single reification will most probably be sufficient.

References

[1] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. Alperin Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks: Exploration in the Representation of Knowledge*. Morgan Kaufmann, San Mateo, CA, 1991.

[2] C. J. Fillmore. The case for the case. In E. Bach and R. Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart and Winston, New York, 1968.

[3] A. Kobsa. The SB-ONE knowledge representation workbench (extended version). Memo 50. SFB 314: AI - Knowledge-Based Systems, Dept. of Computer Science, Univ. of Saarbrücken, Saarbrücken, Germany, 1990.

[4] A. Kobsa. First experiences with the SB-ONE knowledge representation workbench in natural-language applications. *SIGART Newsletter*, Summer, 1991.

[5] A. Kobsa. Reification in meta-SB-ONE: Bridging the object/relation dichotomy. Unpublished Manuscript, Dept. of Computer Science, Univ. of Saarbrücken, Saarbrücken, Germany., 1991.

[6] A. Kobsa. Utilizing knowledge: The components of the SB-ONE knowledge representation workbench. In J. Sowa, editor, *Principles of Semantic Networks: Exploration in the Representation of Knowledge*. Morgan Kaufmann, San Mateo, CA, 1991.

[7] A. Kobsa. The SB-ONE knowledge representation workbench. *Computational Intelligence*, (submitted), 1992.

[8] H. Trost and I. Steinacker. The role of roles: Some aspects of real world knowledge representation. In *Proc. of the 7th IJCAI*, pages 237–239, Vancouver, Canada, 1981.

Statement of Interest, Dagstuhl Workshop

Robert MacGregor
USC/Information Sciences Institute
Marina del Rey, California, United States
macgreg@isi.edu

Below, I list a few topics that are of particular interest to me, and which I would like to be the subject of discussion during the workshop.

1 Semantic Choice Points

The recently circulated KRSS specification emanating from a US-based knowledge representation standards effort, and a proposal for a terminological logic emanating from a group at DFKI, Kaiserslautern represent initial attempts to bring some kind of order to the growing field of terminologically-based systems. I am hoping that, among other things, these specifications will inspire discussion of a number of semantic issues that previously have been ignored for the most part by the community, probably because too few systems had reached a point where they had to bite the bullet and make a particular semantic choice. Below, I list some of these exemplary issues. In each case, we have made an explicit design decision in LOOM (i.e., we have lots of bullet fragments lying around).

1. **Temporal semantics**. Any system that supports either role closure or retraction implicitly defines some sort of temporal semantics. This semantics ought to be made explicit. The need for an explicit semantics becomes even more evident when behavioral constructs such as production rules are introduced into the representational framework.

2. **Retraction semantics**. A number of systems implement retraction of facts. From discussions I've had with Peter Patel-Schneider and Ron Brachman, it appears that LOOM and CLASSIC have adopted a very similar semantics. It would be valuable to survey each of the systems represented at the Workshop to find out if there is in fact a consensus on retraction semantics, or if significant differences of opinion exist.

3. **Skolem individuals**. The generation of objects representing skolem individuals might be considered simply as an implementation detail not deserving of attention by a knowledge level language specification. However, we have users that have explicitly requested that LOOM generate skolem objects in certain situations. Hence, I would like to explore the question of when the system should support retrieval of skolem individuals.

4. **Prototypes, Default Rules, Close-world assumption.** A decade ago it was quite common for KR systems to support such things as generic or prototypical individuals, and default values. Possibly because a formal semantics for such constructs is hard to come by, their implementation has somewhat fallen out of favor. However, it is possible that some sort of consensus could be reached regarding an incomplete characterization of the semantics for these constructs. For example, one might be able find a consensus on the behavior of default rules in the absence of explicit contradictions. Also, LOOM users routinely employ LOOM's facility for specifying that a (non-monotonic) closed-world assumption (predicate completion) should apply to specific relations. The semantics for this in the absence of contradictions appears to be straightforward.

5. **Reified Relations** The PENMAN group at ISI needed a means for specifying concepts representing the reification of binary relations (roles), which resulted in the inclusion of a `defreified-relation` construct into the LOOM language. Subsequent discussions have revealed that the current LOOM semantics for reified relations is not satisfactory. We now have an alternative proposal, which may or may not be controversial.

2 Hybrid Logics

LOOM integrates a description language with a Horn logic [Mac91]. Users have found this combination to be much more useful than having just a bare description language. LOOM has made several decisions regarding control of deductive inference over this hybrid logic that appear to be novel. LOOM offers users the choice of marking their concepts as "forward-chaining" or "backward-chaining." We are finding that for large, dense applications, we can't get acceptable performance if all concepts are marked as forward. Furthermore, LOOM is unable to truth-maintain the more complex concept descriptions, and hence automatically treats these as backward-chaining. Users are requesting such things as lazy evaluation of concept instantiations, combined with caching whenever evaluation actually occurs.

We are hoping that we can discover more high-level means for determining the direction of inferencing. As one example of a more semantically-motivated type of control, LOOM permits concepts to be marked as "monotonic"—this reduces the amount of work necessary to truth maintain these concepts.

We view the object-centered style of representation found in the terminological logics as being antithetical to the relation-based style of representation found in languages such as Prolog. For this reason, I am skeptical of efforts to combine a description language with Prolog (unless Prolog is regarded merely as the host language, rather than as an extension to the representational logic). However, recent efforts to combine feature structures with definite clause grammars appear to have found more successful means for integrating these otherwise dissonant representational paradigms. An exploration at the knowledge level of how such things as features and function symbols within one of these languages map into the terminological framework (definitions, instances, etc.) might prove to be illuminating.

3 Representational Building Blocks

I observed within the DFKI proposal the existence of what might be called "extended numeric restrictions", e.g., `(atleast 1 children Female)`. In languages that support the definition of range-restricted roles, these constructs represent syntactic sugar-coating (e.g., the description `(atleast 1 (and children (range Female)))` is equivalent. However, the (former) extended form has an internal analogue (a data structure), representing what might be called a "type-restricted, numerically-quantified skolem" that appears to me to support more flexible and efficient reasoning than the traditional representations for restrictions. In this case, the extended restriction avoids the necessity for generating the additional role "`(and children (range Female))`" and hence leaves us with a cleaner, more efficient role hierarchy.

The LOOM implementation defines a variety of data structures that we collectively refer to as *features*. These include numeric, type, and filled-by restrictions, role-equivalence descriptions, finite sets, and numerically-bounded intervals (and whenever we get around to implementing them, SDs). Features constitute the building blocks LOOM uses to define its concepts, and as such constitute a vocabulary that is much richer than the vocabularies manipulated by, say, a typical resolution theorem prover. Conceptually, we like to view this as a CISC approach rather than a RISC approach to deductive reasoning. We conjecture that the CISC approach is more amenable to the integration of a large variety of special purpose reasoners than the RISC approach.

References

[Mac91] Robert MacGregor. Using a Description Classifier to Enhance Deductive Inference In *Proceedings of Seventh IEEE Conference on Artificial Intelligence Applications*, Miami, Florida, February 1991.

Large Knowledge Base Management

Eric Mays
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
EMays@Watson.IBM.COM

The focus of our work has been on the systems aspects of terminological languages. In this regard, we have limited the core of our representation to a tractable component. In this core representation there are many things which are unexpressible, and there are some things which are expressible but need not be explicitly represented. A problem thus arises as to how other computational components can be integrated with the representation language to form a useful system. This requires accomodating arbitrary data and functional attachment, as well as incorporating the semantics of primitive types in an extensible way. Additionally, there are performance aspects relating to the scalability of knowledge bases containing tens or hundreds of thousands of concepts, which are required to meet the expectation of current application demands.

Over the past few years we have been involved in building a large knowledge based system, called FAME, which assists IBM marketing representatives in the design of acquisition solutions for large scale computing requirements. The central representation in FAME is a terminological based knowledge representation system called K-Rep, which we have built. Our experience in building a large knowledge based system has shown that there is a need for a knowledge base management system (KBMS) which would support shared access to a large persistent knowledge base by multiple applications. Such a KBMS would support knowledge engineers in the development and maintenance of the knowledge base. The goals for a KBMS consist of the following: (i) to allow a knowledge engineer to update a knowledge base and have these updates persist on secondary storage, (ii) to allow multiple knowledge engineers to have shared access to a knowledge base and modify the knowledge base concurrently, and (iii) to maintain consistency of the shared knowledge base as it evolves.

Our approach to this problem is to adopt a version oriented concurrency protocol in which each knowledge engineer makes modifications to the shared knowledge base, thus deriving multiple versions. We have developed storage management mechanisms which allow any version of the knowledge base to be efficiently updated and retrieved. The version oriented protocol handles the problems relating to long transaction times and large lock grain sizes. Additionally, it places no temporal dependencies on the updates arriving from multiple knowledge engineers. From the multiple versions of the knowledge base it is necessary that a single unified knowledge base emerge. This requirement is achieved via the merge operation.

Additionally, we have been interested in interfacing knowledge representation systems with data bases. One problem which arises here is that the mapping between the data base and the knowledge base can be difficult due to the fine distinctions that are often made in the knowledge base. Terminological logics assist in this regard by allowing the introduction of new concepts in the knowledge base without the necessity for a corresponding modification the the data base schema.

References

[1] C. Apté, R. Dionne, J. Griesmer, M. Karnaugh, J. Kastner, M. Laker, E. Mays, "An Experiment in Constructing an Open Expert System using a Knowledge Substrate", to appear **IBM Journal of Research and Development.**

[2] E. Mays, C. Apté, J. Griesmer, J. Kastner, "Organizing Knowledge in a Complex Financial Domain", **IEEE Expert** 2, Fall 1987, pp. 61-70.

[3] E. Mays, S. Lanka, B. Dionne, R. Weida, "A Persistent Store for Large Shared Knowledge Bases", **IEEE Transaction on Knowledge and Data Engineering** 3, 1, March 1991.

The CLASSIC Knowledge Representation System: Implementation, Applications, and Beyond

Deborah L. McGuinness
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974

Abstract

Implementation, analysis, and application work with CLASSIC have provided opportunities for evaluating the usefulness and implications of our selection of term constructors. We have discovered that all of our applications depend critically on one or more constructs not found in some systems based on terminological logics. We report on user needs (and demands) for sets, individual fillers in concept descriptions, coreference constraints, host language escapes, and simple rules. We discuss some of the advantages and complications that these features introduced from the perspective of both system designers and knowledge engineers.

1 Introduction

The CLASSIC knowledge representation system [1, 2] has been designed and implemented at AT&T Bell Laboratories over the last few years. Within the class of "terminological" knowledge representation systems, it has taken a position in exploring the tradeoff between expressiveness and tractability. While some systems have had a primary goal of extending expressiveness, CLASSIC has tried to remain a simple compositional language that is expressive enough for certain classes of applications while retaining more of a handle on tractability and predictability. This puts it in the same philosophical class as systems such as KANDOR [6] and BACK [8]. CLASSIC's main practical goal has been to provide an implementation that can be used in the real world for certain classes of applications—i.e., it must provide added value over other available tools while retaining understandability and ease of use for non-expert users. Given these goals, CLASSIC provides an interesting testbed for exploration into minimum sets of term constructors. In this paper, we will focus on the ones that may not be considered standard in other terminological systems.

Although work is continuing on the second version of CLASSIC, most of our experience with users has been with the first version so that is the version that will be discussed here. COMMON LISP and C versions of the first release exist and have been in use for approximately two years. The COMMON LISP implementation has been distributed to over 25 Universities and has also been used for several graduate AI classes.[1] In addition, AT&T

[1]In order to request a copy of the COMMON LISP version and user's manual [9], send a letter to

Bell Laboratories has been teaching a class on knowledge representation and CLASSIC for the last year.

The rest of the paper will assume familiarity with the core constructors of most terminological systems (for more background see [12] or [4]), give a very brief introduction to CLASSIC, and discuss the constructors that CLASSIC includes that many other terminological logic systems do not. We will highlight the users' needs for these constructors and discuss the implementation implications. Finally, we will indicate future directions of our work.

2 CLASSIC Overview and Perspective

A simple way to view CLASSIC is KANDOR without the role hierarchy[2] but with rules, tests, coreference constraints, sets, and individual fillers. Roles are atomic but there is a distinction between multi-valued and single-valued roles. In addition to the standard creation of roles, concepts, and individuals, CLASSIC allows information to be retracted from and added to individuals and rules to be retracted from and added to concepts. Concepts are non-circular however rules can create circularities by referring to other concepts.

The representational capabilities of CLASSIC can best be seen by looking at the concept description language. The grammar follows:

```
<concept-description> ::=
   THING | CLASSIC-THING | HOST-THING |
   (built-in host concepts) |
   <concept-name> |
   (AND <concept-expr>+) |
   (ALL <role-expr> <concept-expr>) |
   (AT-LEAST <positive-integer> <role-expr>) |
   (AT-MOST <non-negative-integer> <role-expr>) |
   (SAME-AS <attribute-path> <attribute-path>) |
   (TEST-C <fn> <argument>*) |
   (TEST-H <fn> <argument>*) |
   (ONE-OF <individual-name>*) |
   (PRIMITIVE <concept-expr> <index>) |
   (DISJOINT-PRIMITIVE <concept-expr>
        <group-index> <index>) |
   (FILLS <role-expr> <individual-name>+)
<concept-name> ::=     <symbol>
<individual-name> ::= <symbol> | <cl-host-expr>
<role-expr> ::=        <mrole-expr> | <attribute-expr>
<mrole-expr> ::=       <symbol>
<attribute-path> ::=   (<attribute-expr>+)
<attribute-expr> ::=   <symbol>
```

D. McGuinness at the above address. This should be on university letterhead and it should state that CLASSIC will be used for research and/or educational purposes only.

[2]A primitive role hierarchy will be included in the next release.

```
<cl-host-expr> ::=      <string> | <number> |
                        '<CommonLISP-expr> |
                        (quote <CommonLISP-expr>)
<index> ::=             <number> | <symbol>
<group-index> ::=       <number> | <symbol>
<fn> ::= a three-valued logical function in the
         host language (Common LISP)
<argument> ::= an expression passed to the test function
```

3 Additions and Challenges

We will now discuss five features of CLASSIC that make it much more usable than a system such as KANDOR. One interesting point about these features is that all of our users are critically dependent on one or more of them. We will discuss their usefulness and the problems they present.

3.1 Rules

CLASSIC has a simple forward-chaining rule mechanism. Descriptions can be attached to concepts as rule consequents and, when an individual is known to be an instance of the antecedent concept, the information in the consequent is added to the individual. Rules are used to represent properties that are not used for recognition. For example, one could attach a rule to a concept for person stating that all persons have social security numbers. Then we would not need to know that something has a social security number in order to recognize that this object is a person, but once the object is found to be a person then the object would also become an instance of something that has a social security number. This rule is enforced forever and CLASSIC would signal an error if at any point in the future something was found to contradict the rule. This facility has been found to be very useful in all of our applications.

Rules clearly provide a level of functionality that all of our knowledge engineers desire, yet they have also been the area where most of the application debugging time has been spent. One confusion is that CLASSIC rules are not logical implications. If CLASSIC can prove that something does not have a social security number, then it does not imply that this thing is not a person. Another possibly counter-intuitive notion is that rules are only invoked on individuals. Thus, a concept defined as "person with social security number" would not subsume a concept person with a rule requiring all instances to have social security numbers attached to it. For more on these representational issues see [2]. Another aspect of rules arises when rules that reference other concepts are added to existing concepts. This is the only place in CLASSIC where cycles can be created. This actually provides much more expressive power and does not cause termination problems but it does sometimes create debugging problems. The problem is not that we can't explain the idea of rules, it is just that in some cases CLASSIC's use of rules is different than the rules that some users may have been exposed to in logic or in expert systems. We could probably decrease some of the expectation mismatches if we used rules in concept classification and treated them as logical implications but then we would introduce the problems associated with cycles [5] and require reasoning by cases. One other possibility is to adopt the OPS method of handling rules but then part of the power of rules for integrity checking is lost since rules would not continually enforced.

3.2 Coreference Constraints

The **SAME-AS** constructor requires that the two composed attribute (uni-valued role) paths have the same filler. The original motivation for this came from natural language and planning uses where it is important for the actor of one act to be the recipient of another. Other obvious uses for this are in layout where it is important that two ends of a wire are attached to the same wire.

Difficulties with **SAME-AS** have not really arisen for the application programmer as they did with tests—the difficulties show up in the implementation. Our original plan was to implement coreference constraints on multivalued roles. We did not have a complete theoretical analysis of the task when we began, and midway through our implementation (while we were still struggling with the algorithm), a proof was provided (see [10] and [7]) showing the reason for our problems. Even after limiting **SAME-AS** to functional roles, we still found this portion of the code to be the most challenging to write and maintain. It also is the constructor that is the most challenging when considering extensions to the system.

3.3 Tests

By using tests, users may write functions that determine membership in a concept class. The function could be something as simple as checking for an even number of fillers on a role or it could be arbitrarily complex host code (either COMMON LISP or C depending upon the implementation). Tests have proven to be invaluable in applications. In limited languages it can be quite important to have a way for the user to add a few specialized concepts to the application. Also, given that certain kinds of expressive power have been deliberatively left out of version 1 in an attempt to be more tractable, tests can provide a mechanism for expressing things that are outside CLASSIC's scope but still essential to the application. One common test of expressive power and reasoning in terminological systems is the "at least one child who is a doctor" issue. CLASSIC allows only number restriction (at least one child) and value restriction (all children are doctors) without using tests. One can write a test function to recognize individuals who have at least one child who is a doctor. This solves the expressiveness and recognition issue but it doesn't attack the part of this problem that Brachman and Levesque showed to be intractible [3].

Tests are treated as black boxes by CLASSIC and originally we thought that they would not interact with the rest of the CLASSIC code. After we required tests to be three-valued monotonic side-effect-free functions, we succeeded in allowing tests to coexist harmoniously with the rest of the system. From a terminological system designer's perspective, tests can provide a wonderful view into the real needs of a user. Most users will minimize their use of tests so analyzing the final uses of tests has been instructive. Version 2 of CLASSIC has been expanded in two ways (intervals and role hierarchies) as a result of evaluating user's test usage.

Tests and rules together can provide a powerful combination. Rules are limited to a form that uses one named concept expression as an antecedent and another concept expression as the consequent. If a rule is attached to a test concept, then this rule can have arbitrary expressive power in its antecedent. Also, if the consequent concept expression includes a test concept, then the rule can provide more expressive power in an integrity checking mode.

3.4 Sets

The **ONE-OF** constructor allows the user to say that all of the fillers for a role must be in a specified set of individuals. For example, a wine's color must either be red, white, or rose. **ONE-OF** provides both a limited form of negation (i.e., the wine is not blue) as well as a kind of disjunction, and it has been quite useful in expressing the incomplete knowledge that seems to pervade most of our applications.

ONE-OF, however, has introduced a form of incompleteness into our system. Since **ONE-OF** is inherently a form of disjunction, in order to reason completely with it one must do reasoning by cases. CLASSIC specifically does not attempt any such work and is thus incomplete. **ONE-OF**, as **FILLS** below, allows concept descriptions to refer to individuals. This presents some questions when calculating the subsumption hierarchy. In normal evolution of a knowledge base (ignoring errors for the moment), we expect some things to change about individuals (for example, a person might change from being single to being married or might become a parent), but we don't expect definitions of concepts (such as person or parent) to change. Thus, we expect the concept hierarchy to remain constant while "contingent facts" about individuals may change. Concept descriptions that include individuals in their definition fall somewhere in the middle since although they are clearly concepts, something that they refer to may change. CLASSIC handles this by doing concept subsumption without taking into account the contingent facts about individuals. Thus a concept that had a role filled by "**ONE-OF** Joe or John" could be subsumed by something that had at most 2 fillers for that role (since the fact that Joe and John are unique is not contingent) but it would not be subsumed by a concept that expects this role to be filled by New Jersey residents even if Joe and John are both currently known to live in New Jersey (since one or both of them could move). This issue is similar to the one with rules—the behavior of CLASSIC is well motivated and explainable, but sometimes runs counterintuitive to new users' expectations and thus can be a source of confusion.

3.5 Fills

In some senses, **FILLS** can be viewed as a special case of **ONE-OF**. Instead of saying that the filler of a role belongs to some set of individuals, **FILLS** states that one particular individual fills the role. The difference between **FILLS** and **ONE-OF** is that **ONE-OF** implies number restrictions, i.e., the maximum number of fillers of a role is equal to the number of elements in the **ONE-OF** set, while **FILLS** just states that one of the fillers of that role is known but (in the absence of other number restrictions) other fillers can be added to the role later. Since **FILLS** is like **ONE-OF** when it allows individuals to be referenced in concept expressions it has some of **ONE-OF**'s advantages and difficulties.

4 Current and Future Research

Our experience with theoretical analysis of terminological systems, implementation of CLASSIC, and its use in several kinds of applications continues to drive our work forward. We are currently designing the next version of CLASSIC. While we are still exploring the space of small usable systems, the next version will be richer in expressive power as well as in supporting facilities (such as explanation, dumping knowledge bases, UNIX inspired line-oriented interface, graphical interface, and query language). We are also pursuing research on a version of CLASSIC that will be extensible given templates designed by the user to define additional inferences that CLASSIC should perform.

References

[1] Alex Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 59–67, June 1989.

[2] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori Alperin Resnick, and Alex Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In Sowa [11].

[3] Hector J. Levesque and Ronald J. Brachman. A fundamental tradeoff in knowledge representation and reasoning (revised version). In Ronald J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation*, pages 42–70. Morgan Kaufmann, San Mateo, California, 1985.

[4] Robert MacGregor. The evolving technology of the classification-based knowledge representation systems. In Sowa [11].

[5] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In Sowa [11].

[6] Peter F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pages 11–16, Denver, Colorado, December 1984. IEEE Computer Society. A revised and extended version is available as AI Technical Report Number 37, Schlumberger Palo Alto Research, October 1984.

[7] Peter F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2):263–272, June 1989.

[8] Christof Peltason, Albrecht Schmiedel, Carsten Kindermann, and Joachim Quantz. The BACK system revisited. KIT-Report 75, Department of Computer Science, Technische Universität Berlin, August 1989.

[9] Lori Alperin Resnick, Alex Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Peter F. Patel-Schneider. CLASSIC description and reference manual for the COMMON LISP implementation. AT&T Bell Laboratories., January 1990.

[10] Manfred Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 421–431. Morgan Kaufmann, May 1989.

[11] John Sowa, editor. *Principles of Semantic Networks: Explorations in the representation oj knowledge.* Morgan-Kaufmann, San Mateo, California, 1991.

[12] William A. Woods and James G. Schmolze. The KL-ONE family. Harvard Technical Report Number TR-20-90, Harvard University, August 1990.

The Complexity of Concept Languages – Extended Abstract –

Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi
Università di Roma "La Sapienza"
via Salaria 113, I-00198, Roma, Italy

Werner Nutt
German Research Center for Artificial Intelligence (DFKI)
Postfach 2080, D-6750 Kaiserslautern, Germany

Concept languages have been investigated mainly in the field of knowledge representation, following the ideas initially embedded in many frame-based and semantic-network-based languages, especially the KL-ONE language [BS85]. However, several formalisms of this kind are now being considered within the realm of data bases and logic programming, with the aim of enriching the expressivity of existing data models and logic programming languages with object-oriented features (see [Be88, AN86]).

Concept languages can be given a Tarski style declarative semantics that allows them to be conceived as sublanguages of predicate logic [BL84]. A concept is built up of two kinds of symbols, primitive concepts and primitive roles. An interpretation interprets them as subsets of a domain and binary relations over the domain. These primitives can be combined by various language constructs (such as intersection, union, role quantification, etc.) yielding complex concepts, which again are interpreted as subsets of the domain. Different languages are distinguished by the different sets of constructs they provide.

To give examples we suppose that person and female are primitive concepts, and child and female_relative are primitive roles. Using the set theoretical connectives intersection and complement, we can describe the class of "persons that are not female" by the concept

$$\mathsf{person} \sqcap \neg\mathsf{female}.$$

Most languages provide quantification over roles that allows for instance to describe the classes of "individuals whose children are all female" and "individuals having a female child" by the concepts

$$\forall\mathsf{child}.\mathsf{female} \quad \text{and} \quad \exists\mathsf{child}.\mathsf{female}.$$

Number restrictions on roles denote sets of individuals having at least or at most a certain number of fillers for a role. For instance,

$$(\geq 3\,\mathsf{female\_relative}) \sqcap (\leq 2\,\mathsf{child})$$

denotes the class of "all individuals having at least three friends and at most two children." Intersection can also be used as a role forming construct. For instance, the intersection

$$\mathsf{child} \sqcap \mathsf{female\_relative},$$

intuitively yields the role "daughter."

The basic reasoning tasks on concepts are satisfiability and subsumption checking. A concept is unsatisfiable if it denotes the empty set in every interpretation, and is satisfiable otherwise. A concept C is subsumed by a concept D if in every interpretation C denotes a subset of the set denoted by D . For a long time, the KL-ONE community was content with sound, but incomplete subsumption algorithms. Such an algorithm delivers a correct answer when given C and D such that C is not subsumed by D, but sometimes fails to recognize that one concept is subsumed by another one.

Complexity analysis of the subsumption problem originated with the paper [BL84] by Brachman and Levesque, which provides a polynomial algorithm for a very limited language, called $\mathcal{FL}^-$, and shows that for the seemingly slightly more expressive language $\mathcal{FL}$ subsumption is co-NP-hard. Nebel [Ne88] identified other constructs that give rise to co-NP-hard subsumption problems. Neither [BL84] nor [Ne88] give algorithms for the co-NP-hard languages. The first nontrivial subsumption algorithm was devised by Schmidt-Schauß and Smolka [SS91] for the language $\mathcal{ALC}$, an extension of $\mathcal{FL}$. They proved that unsatisfiability and subsumption in $\mathcal{ALC}$ are PSPACE-complete and identified a sublanguage with co-NP-complete unsatisfiability problem.

In the present paper, we consider a family of languages, called $\mathcal{AL}$-languages, which includes most of the concept languages considered in the literature. In the simplest $\mathcal{AL}$-language, called $\mathcal{AL}$, *concepts* (denoted by the letters C and D) are built out of *primitive concepts* (denoted by the letter A) and *primitive roles* according to the syntax rule

$$C, D \longrightarrow A \mid \top \mid \bot \mid C \sqcap D \mid \neg A \mid \forall R.C \mid \exists R.\top$$

where R denotes a *role*, that in $\mathcal{AL}$ is always primitive (more general languages provide a constructor for role intersection).

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a set $\Delta^\mathcal{I}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^\mathcal{I}$ (the *interpretation function* of $\mathcal{I}$) that maps every concept to a subset of $\Delta^\mathcal{I}$ and every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that

$$\begin{aligned}
\top^\mathcal{I} &= \Delta^\mathcal{I} \\
\bot^\mathcal{I} &= \emptyset \\
(C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(\neg A)^\mathcal{I} &= \Delta^\mathcal{I} \setminus A^\mathcal{I} \\
(\forall R.C)^\mathcal{I} &= \{a \in \Delta^\mathcal{I} \mid \forall b. (a,b) \in R^\mathcal{I} \rightarrow b \in C^\mathcal{I}\} \\
(\exists R.\top)^\mathcal{I} &= \{a \in \Delta^\mathcal{I} \mid \exists b. (a,b) \in R^\mathcal{I}\}.
\end{aligned}$$

Obviously, an interpretation function is already determined by the way it interprets primitive concepts and roles. An interpretation $\mathcal{I}$ is a *model* for a concept C if $C^\mathcal{I}$ is nonempty. A concept is *satisfiable* if it has a model and *unsatisfiable* otherwise. We say C is *subsumed* by D if $C^\mathcal{I} \subseteq D^\mathcal{I}$ for every interpretation $\mathcal{I}$, and C is *equivalent* to D if $C^\mathcal{I} = D^\mathcal{I}$ for every interpretation $\mathcal{I}$.

More general languages are obtained by adding to $\mathcal{AL}$ the following constructs:

- *union* of concepts (indicated by the letter $\mathcal{U}$), written as $C \sqcup D$, and defined by
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}};$$

- full *existential quantification* (indicated by the letter $\mathcal{E}$), written as $\exists R.C$, and defined by
$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\};$$

- *complement* of non-primitive concepts (indicated by the letter $\mathcal{C}$), written as $\neg C$, and defined by
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}};$$

- *number restrictions* (indicated by the letter $\mathcal{N}$), written as $(\geq n\, R)$ and $(\leq n\, R)$, where n ranges over the nonnegative integers coded in unary (i.e., the integer n is represented by a string of length n), and defined by
$$(\geq n\, R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \geq n\},$$
and
$$(\leq n\, R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \leq n\},$$
respectively;

- intersection of *roles* (indicated by the letter $\mathcal{R}$), written as $Q \sqcap R$, where Q and R are arbitrary roles, and defined by
$$(Q \sqcap R)^{\mathcal{I}} = Q^{\mathcal{I}} \cap R^{\mathcal{I}}.$$

We consider all combinations of the above constructs in concept languages. Every $\mathcal{AL}$-language is named by a string of the form
$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{R}],$$
where a letter in the name stands for the presence of the corresponding construct in the language. Observe that the combination of union and full existential quantification gives the possibility to express complements of concepts, and conversely, union and full existential quantification can be expressed using complements. Hence, without loss of generality we will assume that union and full existential quantification are available in every language that contains complements, and vice versa. In language names we will use the letter $\mathcal{C}$ instead of $\mathcal{UE}$. It follows that there are 16 pairwise non-equivalent $\mathcal{AL}$-languages, which form a lattice, whose bottom element is $\mathcal{AL}$ and whose top element is $\mathcal{ALCNR}$.

The present paper features two main results. First, we define a general technique for checking unsatisfiability (and therefore subsumption, since C is subsumed by D if and only if $C \sqcap \neg D$ is unsatisfiable) in $\mathcal{AL}$-languages, thus providing complete algorithms for the basic inferences in concept languages. Following an idea presented in [SS91], our technique relies on a set of rules, which closely resemble the rules of the tableau calculus for first order logic. In fact, if one translates concepts into predicate logic formulas, and applies to them the tableaux calculus with a suitable control strategy, one obtains essentially the calculus described here.

Second, we give a detailed complexity analysis of both unsatisfiability and subsumption for $\mathcal{AL}$-languages. We have classified 15 of the 16 languages with respect to the complexity of both problems. The only exception is $\mathcal{ALEN}$, where we can say that unsatisfiability and subsumption are in PSPACE, and both are co-NP-hard (this follows from the results reported in [Ne88]) and NP-hard. Notice that for only one of these languages, namely $\mathcal{ALC}$, both the upper and the lower complexity bound for the two problems were previously known. Since for all $\mathcal{AL}$-languages but one the upper and lower bounds we give coincide, one can say that our algorithms are optimal for the problems they solve.

Complete proofs of the results are given in [DL*91].

References

[AN86] H. Aït-Kaci, R. Nasr. "LOGIN: a logic programming language with built-in inheritance." *Journal of Logic Programming,* 3, 1986.

[Be88] C. Beeri. "Data models and languages for databases." *Proc. ICDT-88,* 1988.

[BL84] R. J. Brachman, H. J. Levesque. "The tractability of subsumption in frame based description languages." *Proc. 4th AAAI,* Austin, Tex., 1984.

[BS85] R. J. Brachman, J. Schmolze, "An overview of the KL-ONE knowledge representation system." *Cognitive Science*, 9 (2), 1985.

[DL*91] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt. *The Complexity of Concept Languages.* DFKI Research-Report, Forthcoming.

[GJ79] M. R. Garey, D. S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, Cal., 1979.

[HN*90] B. Hollunder, W. Nutt, M. Schmidt-Schauß, "Subsumption algorithms for concept description languages." *Proc. 9th ECAI, 1990.*

[Ne88] B. Nebel. "Computational complexity of terminological reasoning in BACK." *Artificial Intelligence*, 34(3), 1988.

[Ne90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, Springer LNAI 422, 1990.

[SS91] M. Schmidt-Schauß, G. Smolka. "Attributive concept descriptions with complements." To appear in *Artificial Intelligence*, **47**, 1991.

Position Paper - Terminological Logics Workshop

Lin Padgham
Linköping University
Department of Computer and Information Science

My main work has been on the development of a logic for default reasoning within type hierarchies. Unlike most people I have worked with a notion of 2 parts (or nodes) for each type descriptor - one with necessary and one with default information. This gives similar expressivity to systems with mixed link types (strict and default links) with some interesting extra expressivity (one can say e.g. that a sparrow is a typical bird, as well as that a typical bird is a flying thing). This approach has also enabled development of an algorithm for default reasoning which is essentially linear. I have a prototype implementation of the default reasoner.

The model for default reasoning is a lattice based model where the core and default nodes of types, and also the objects can be compared to each other with respect to a hypothetical lattice of feature descriptors. This is equivalent to all inheritance links being strict. The fact that all inheritance links are strict in our model (default reasoning is accomplished by a "jump" against the direction of the link, between type core and type default) makes the model relatively compatible with models used by terminological logic systems. This leads to the hope that we can use our theory of default reasoning as a base for specifying well behaved default reasoning within terminological logic systems. The representation of the type hierarchy for our default reasoner is, as mentioned previously based only on strict inheritance. Consequently it is possible to use classification algorithms developed within terminological logics on this hierarchy.

Together with a graduate student (Tingting Zhang) i am working on a medical diagnosis system which uses a combination of default reasoning and classification in the diagnosis process. Disease descriptions are represented in a type hierarchy, where each disease contains a set of necessary and a set of typical features. A patient is represented as a set of symptoms. An initial classification is made of the patient and then more information is sought in order to move the classification down in the hierarchy.

The language used for the diagnosis application is extremely limited compared to the languages usually used in terminological logics. However it appears adequate for this application. We suggest that for some applications it may be desirable to limit the expressiveness of the language in certain ways in order to allow integration of defaults.

Relevant Published Work of Mine

References

[Padgham 90] Padgham, L. Defeasible Inheritance: A Lattice Based Approach, *Computers and Mathematics with Applications*, special issue on Semantic Nets.

[Padgham 89] Lin Padgham, Non-Monotonic Inheritance for an Object-Oriented Database, 1989, ISBN 91-7870-485-5 (PhD thesis).

[Padgham 89b] Padgham, L. Negative Reasoning Using Inheritance, Proceedings of IJCAI '89, Aug 20-25, 1989, Detroit, Michigan, USA. vol.2 p. 1086.

[Padgham 89c] Padgham, L. A Lattice Based Model for Inheritance Reasoning, Proceedings of the Workshop on Inheritance in Programming Languages and Knowledge Representation, Viareggio, Italy, February 6-8, 1989.

[Padgham 88] Padgham, L. A Model and Representation for Type Information and its Use in Reasoning with Defaults. *Proceedings of AAAI '88* St Paul, Minnesota, August 20-26, 1988, vol 2, pp.409-414

[Padgham 88b] Padgham, L. NODE: a database for use by intelligent systems. Proceedings of the International Symposium on Methodologies for Intelligent Systems, Torino, Italy, October 17-19, 1988.

[Zhang, Padgham 90] Zhang T., Padgham L., A Diagnosis System Using Inheritance in an Inheritance Net, *Proceedings of the International Symposium on Methodologies for Intelligent Systems*, Knoxville, USA, Oct 24-27 1990.

[Padgham, Rönnquist 87] Lin Padgham, Ralph Rönnquist, LINCKS: An Imperative Object Oriented System, *Proceedings of the 20th International Conferences on Systems Sciences*, Hawaii, Jan. 1987.

Handling Computational Difficulties with Reasoning in Terminological Logics

Peter F. Patel-Schneider
AT&T Bell Labs, 600 Mountain Avenue
Murray Hill, New Jersey 07974
U. S. A.
pfps@research.att.com

It is an unfortunate fact of life that the important reasoning operations (subsumption, classification, and realization) in many terminological logics are worst-case intractable or even undecidable. (See [2] for some of these results.) Because these operations will be performed often during the normal course of operation of knowledge representation systems built upon terminological logics, I think that some solutions to the problem must be devised before knowledge representation systems based on terminological logics will be of general use.

I see two basic types of solutions to this problem:

1. choosing a better method of analyzing complexity and decidability, and
2. retreating to incomplete (or unsound) reasoning.

Both of these basic types of solutions have several variants, some better than others.

One solution to the problem fits in neither of the two types given above. This solution—limiting the expressive power of the terminological logic to obtain worst-case tractable reasoning—was initially suggested by Brachman and Levesque and partially implemented in KANDOR [5] and CLASSIC [1]. (Neither KANDOR nor CLASSIC are worst-case tractable, complete reasoners for a standard terminological logic, so they are not pure examples of this approach.) However, recent results in the complexity and decidability of reasoning in terminological logics have shown that achieving worst-case tractability requires giving up far too much expressive power, so this solution cannot be considered to be viable, at least by itself.

1 Complexity Analysis

The worst solution in the better analysis camp, in my view, is to simply ignore the problem and hope that actual systems will not have any problems. This may work in some cases, but there are many applications where it is intolerable not to have some idea of how fast (or slow) the knowledge representation will be.

However, worst-case tractability is an unduly pessimistic indicator of performance. If it could be guaranteed that the system would in fact perform reasonably quickly in almost all knowledge bases and queries that will be encountered, then the system would be of considerably more use.

This "average-case" complexity analysis suffers from two problems. First, it is very hard to give a probabilistic characterization of the knowledge bases and queries that a knowledge representation system will encounter. Second, there is still no firm guarantee that the system will not take much too long—just a guarantee that this will happen rarely. Many applications can tolerate rare problems of this sort, perhaps by having a method for terminating reasoning if it is taking too long, but others can not.

Another replacement for worst-case complexity analysis is "normal-case" analysis. Here a subset of the possible knowledge bases is selected as "normal", and the system is guaranteed to be tractable on this subset. If the normal cases include all those that will be encountered in an application, then the system is effectively worst-case tractable for this application. Normal-case tractability is generally no worse to determine, given a definition of a normal case, than worst-case tractability is, and is a viable replacement for worst-case tractability.

However, it is often possible to go beyond normal-case tractability by performing a complete computational analysis of a reasoning algorithm. From this algorithmic analysis not only can tractable cases be extracted, but also the causes of exponential behavior can be determined and quantified.

It is the quantification of intractable behavior that is probably the most important benefit of this type of analysis. In the best case, it is possible to provide a bound on how long a particular operation will take just by quickly looking at the form of the inputs. The application can then avoid situations that could possibly result in long operations.

One particular example of this computational analysis is the complexity of subsumption in terminological logics in the presence of definitions. Bernhard Nebel [4] has shown that the presence of definitions results in intractable subsumption in many simple terminological logics. However, in many applications the replacement of names by their definitions just does not result in a large increase in the size of descriptions, and whenever this is the case, computational difficulties will not occur.

The normal-case and computational analysis solutions to the computational problems of terminological logics appear to work fairly well for a number of intractable terminological logics. Unfortunately, it is much harder to perform such magic when operations are undecidable.

2 Incomplete (and Unsound) Reasoning

The second type of solution to the complexity problem for terminological logics is to give up on sound and complete reasoning in return for computational benefits.

The traditional solution to complexity problems has been to just implement those deductions that can easily be implemented and that produce a reasonably fast system, or that are needed to make particular applications (or demos) work. As more and more deductions are added, it becomes harder and harder to determine just what deductions these systems do perform, and harder and harder to count on the system. I consider this an inferior solution, even in combination with other solutions.

I see three good methods for describing the deductions of a partial reasoner. They are to use

1. a non-standard semantics for the logic, such as my four-valued semantics [6] or the non-standard semantics used in CLASSIC;
2. a set of inference rules, such as the rules in [7] and [2]; or
3. an abstract algorithm.

Each of these solutions, when done well, can result in a description of the reasoner that can be readily understood and used to develop applications.

Non-standard semantics can describe both radical and minor changes to reasoners with a single modification in the semantics, as in the non-standard semantics for CLASSIC's subsumption algorithm. CLASSIC allows user code to appear in concept descriptions, which, obviously, makes determining subsumption undecidable. The non-standard semantics for subsumption in CLASSIC treats user code as arbitrary functions, divorced from the semantics of the programming language. This sanctions treating user code as black boxes. CLASSIC also allows individuals to appear in concept descriptions, which makes subsumption worst-case intractable. The non-standard semantics for CLASSIC maps individuals into subsets of the domain instead of elements of the domain. This means that certain deductions are not valid for individuals, removing a source of intractability. However, developing non-standard semantics with the correct computational properties is a difficult process at best, and not a short-term solution.

I think that a good description of incomplete terminological reasoners that are likely to be built in the near future can be obtained by means of an abstract algorithm. In particular, an abstract algorithm description of the standard normalize-and-classify method used in CLASSIC, LOOM [3], KANDOR, etc., should be quite easy for users to understand. Sets of inference rules, although easy to devise and modify, can be very hard to understand, even for experts.

Perhaps the best method for describing incomplete terminological reasoners will be found where abstract algorithms and structured sets of inferences rules merge. Here it may be possible to obtain the benefits of both algorithms (easy-to-follow control flow) and inference rules (independence from many low-level details).

3 Combining Solutions

I think that just about any system that attempts to solve the computational difficulties inherent in reasoning in terminological logics will have to use several solutions. For example, CLASSIC obtains tractable subsumption (without definitions) by implementing a partial subsumption algorithm that is sound and complete with respect to a variant semantics for an expressively limited terminological logic but also uses the computational analysis method to characterize its intractability with respect to definitions. More work is needed to determine just what is the best sort of description for such combination solutions.

References

[1] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the representation of knowledge.* Morgan-Kaufmann, San Mateo, California, 1991.

[2] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 151–162. Morgan Kaufmann, May 1991.

[3] R. M. MacGregor and R. Bates. The Loom knowledge representation language. Technical Report ISI/RS-87-188, Information Sciences Institute, University of Southern California, May 1987.

[4] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249, May 1990.

[5] P. F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pages 11–16, Denver, Colorado, Dec. 1984. IEEE Computer Society. A revised and extended version is available as AI Technical Report Number 37, Schlumberger Palo Alto Research, October 1984.

[6] P. F. Patel-Schneider. A four-valued semantics for terminological logics. *Artificial Intelligence*, 38(3):319–351, Apr. 1989.

[7] M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with unions. *Artificial Intelligence*, 47, 1991.

Modeling and Reasoning

Joachim Quantz
Technical University of Berlin
Project KIT-BACK, FR 5-12
Franklinstr. 28/29, W-1000 Berlin 10, Germany
E-Mail: quantz@tubvm.cs.tu-berlin.de

Modeling and reasoning are, of course, strongly connected to the expressivity of the representation language under investigation. Thus, when considering the integration of new epistemological primitives, one tries to find out what can be modeled with these primitives and what kind of inferences are licensed by them. From a theoretical point of view, once the semantics of a primitive is specified, all inferences are determined as well. From an applicational point of view, however, it is important to characterize those inferences which can be regarded as obvious or as basic. This distinction between obvious and more elaborated inferences is useful for two reasons. If, for whatever reason, only an incomplete algorithm is implemented, it should at least compute the obvious inferences. Furthermore, even for complete algorithms, obvious inferences might be efficiently computable whereas more elaborated ones might be not. In this case one could choose to precompute obvious inferences, whereas other inferences are only computed on demand by the user.[1]

As a result of my research concerning the integration of rules (or implication links) and role forming operators into BACK,[2] I became convinced, that the obvious inferences and hence the advantages of these extensions lie on the object level. In the following I want to substantiate this claim with respect to role forming operators by summarizing some of the results presented in [4].[3]

Role Forming Operators

Research in the area of terminological representation systems tends to focus on concept forming operators and on concept subsumption, whereas role forming operators and role subsumption are comparatively neglected. Out of 11 systems listed in an overview in [3, S. 234] 4 do support primitive role hierarchies and 5 even include role forming operators for the definition of roles. It is not obvious, however, to what extent the inference algorithms

[1]The distinction made by the system developer is, of course, only preliminary and its adequacy has to be verified when actual applications are evaluated.

[2]This work was supported by the Commission of the European Communities and is part of Esprit Project AIMS (5210).

[3]For a theoretical investigation of rules or implication links confer [6]. A description of the implementation in BACK and a small example illustrating the usefulness of implication links can be found in [5].

of hybrid representation systems have to be modified in order to support the integration of defined roles into a terminological language.[4] Three kinds of consequences can be distinguished:

Concept Subsumption

The use of role forming operators can lead to subsumption between concepts (e.g., **all**(r_1,**atmost**(m,r_2)) **and atmost**(n,r_1) $\preceq$ **atmost**(nm,r_1 **comp** r_2)). To capture these subsumption relations classification algorithms for concept terms have to be modified in a non-trivial way.

Role Subsumption

In addition to classification for concept terms subsumption between role terms (e.g., **trans**(r **and range**(c)) $\preceq$ **trans**(r) **and range**(c)) must be computed by the classifier. If the **trans** operator is not included in the terminological language, the algorithm presented in [4] together with two minor modifications is probably complete and subsumption of role terms is then turing reducible in polynomial time to subsumption of the domain and range concepts. The conjecture that the algorithm is complete for full $\mathcal{TFR}$ turned out to be unjustified however, due to subsumption between embedded **comp** and **trans** terms (e.g. **trans**(r_1) **comp** $r_2 \equiv$ **trans**(r_1) **comp trans**(r_2) iff $r_2 \preceq r_1$).

Hybrid Entailment

Additional inferences on the object level are from my point of view the major contribution of defined roles for a hybrid representation system. In general, the terminological modeling with defined roles can be used to establish complex connections between roles which result in the automatic instantiation of role-fillers on the object level (for examples, see below).[5]

Dependencies

Having distinguished these areas of hybrid reasoning one can ask what kind of dependencies exist between them. Clearly, role subsumption has impact on concept subsumption (via **atleast**, **atmost**, and **all** terms) and vice versa (via **domain** and **range** terms). Also, both concept and role subsumption influence hybrid entailment (e.g., $r_1(o_1,o_2) \models r_2(o_1,o_2)$ if $r_1 \preceq r_2$). There are, however, inferences on the object level which must be drawn by the recognizer alone, without resort to the classifier. The integration of role forming operators gives rise to the following list of genuine recognition tasks:

- Counting of role-fillers:
  $r(o_1,o_2) \wedge r(o_1,o_3) \models$ **atleast**$(2,r)(o_1)$.[6]

[4]In the following I consider the language $\mathcal{TFR}$ containing the role forming operators **domain**, **range**, **and**, **inv**, **comp**, and **trans**.

[5]The importance of role-filler instantiation is underlined in proposals for semantic data models (e.g., SDM [1]). Defined roles and the role forming operators **comp** and **inv** correspond to the 'member attribute interrelationships' in SDM.

[6]Together with classification this can be used for the propagation of domains (**domain**(c)$(o_1,o_2) \models c(o_1)$), since **atleast**$(1,$ **domain**$(c)) \preceq c$.

- Propagation of value restrictions:
  $\mathbf{all}(r,c)(o_1) \wedge r(o_1,o_2) \models c(o_2)$.
- Generalization over closed role-filler sets:
  $\mathbf{atmost}(1,r)(o_1) \wedge r(o_1,o_2) \wedge c(o_2) \models \mathbf{all}(r,c)(o_1)$.
- Instantiation of inverse role-fillers:
  $r(o_1,o_2) \models \mathbf{inv}(r)(o_2,o_1)$
- Instantiation of role-filler chains:
  $r_1(o_1,o_2) \wedge r_2(o_2,o_3) \models r_1\ \mathbf{comp}\ r_2(o_1,o_3)$.[7]

Conclusion

In general, most object level inferences resulting from defined roles are either computable by the recognizer or can be derived via role classification. Thus, even if consequences for concept subsumption are not integrated into the classification algorithm the interesting inferences on the object level are still derivable.[8] This line of argument (ignore consequences for concept subsumption, focus on consequences for hybrid entailment) might be also valid for the use of constants in concept definitions and for role value maps.

Whereas so far only concept valued roles were considered, similar extensions are possible for roles which have special types as their ranges.[9] For these roles additional role forming operators can be provided, such as numerical operations or basic set operations. This might be especially useful for numerical, functional roles. Thus value_added_tax could be defined as net_price * 0.14. This would lead to the automatic instantiation of the value for value_added_tax when the value for net_price is specified.

These considerations suggest a revision of the application scenario for terminological logics. In the traditional application scenario the user enters definitions of concepts and the system detects implicitly given subsumption relations and builds up an explicit terminological hierarchy. As a consequence, a terminological representation system is considered useful only for domains in which defined concepts (necessary and sufficient conditions) can be specified. Furthermore the concept classifier is considered to be *the* inference component whereas recognition is only a special form of classification. In the revised application scenario, however, the modeling of a terminological hierarchy is just a first step followed by the specification of rules. The system supports these activities by checking consistency and eventually by drawing additional conclusions. In a second step object descriptions are evaluated taking into account terminological and rule knowledge. Additional properties of objects can be inferred from these descriptions. In this scenario a terminological representation system can be even useful if the terminology does not contain any defined concepts, since some object level inferences do not involve concept subsumption.[10] Besides the concept classifier there are other important inference compo-

[7]This is also the basic schema for inferences involving the transitive closure of roles.

[8]To compute these inferences a recognition algorithm as presented in [5] has to be augmented. In addition, the conceptual indexing of objects which supports retrieval for both simple and complex queries (cf. [2]) has to be expanded to include indexing of role-fillers.

[9]The current version of BACK supports attribute-sets, numbers, and strings as role ranges.

[10]Needless to say, that the more defined concepts a domain contains, the more useful is a terminological representation system.

nents like the role classifier, the rule classifier and the recognizer. They all use the concept classifier but they also perform interesting inferences on their own.

References

[1] M. Hammer, D. McLeod, "Database Description with SDM: A Semantic Database Model", *ACM Transactions of Database Systems* **6**, 351–386, 1981

[2] C. Kindermann, "Class Instances in a Terminological Framework – an Experience Report", in H. Marburger (ed.), *Proc. of GWAI-90*, Berlin: Springer, 48–57, 1990

[3] B. Nebel, *Reasoning and Revision in Hybrid Representation Systems*, Berlin: Springer, 1990

[4] J. Quantz, *Modeling and Reasoning with Defined Roles in BACK* KIT Report 84, Technical University of Berlin, 1990

[5] J. Quantz, C. Kindermann, *Implementation of the BACK System Version 4*, KIT Report 78, Technical University of Berlin, 1990

[6] K. Schild, *Towards a Theory of Frames and Rules*, KIT Report 76, Technical University of Berlin, 1989

From Terminological Logics to Modal Logics

Klaus Schild
Technische Universität Berlin
KIT-BACK, FR 5-12, Franklinstr. 28/29
W-1000 Berlin 10, FRG

Correspondences between terminological logics and propositional modal and dynamic logics are currently in the focus of my interest.[1] These correspondences turn out to be highly productive because formerly unrelated fields are brought together. In the area of terminological logics, running systems have been developed since the late seventies. Only recently theoretical investigations have been undertaken mainly concerning the computational complexity of terminological logics. In the very contrast to that, elaborated theories for modal and dynamic logics have been developed much earlier. Particularly, for modal logic there is—apart from first order logic—the most elaborated logical theory, and dynamic logic has benefited from these results. By detecting these correspondences, one can gain new insights into terminological logics solely by expounding the theorems of modal and dynamic logic as theorems of the corresponding terminological logic.

Terminological Logic and Modal Logic

The terminological logic $\mathcal{ALC}$, introduced by Schmidt-Schauß and Smolka [9], comprises the Boolean operators $\sqcap$, $\sqcup$, and $\neg$ on concepts as well as the value restrictions $\forall R.C$ and $\exists R.C$.

It is well known that $\mathcal{ALC}$ is a sublanguage of first order logic since atomic concepts correspond to one-place predicates and atomic roles to two-place predicates. The $\mathcal{ALC}$-concept $\neg c_1 \sqcup \forall r.c_2 \sqcap c_3$, for instance, can be expressed by the first order formula $\neg c_1(x) \vee \forall y.r(x,y) \Rightarrow c_2(y) \wedge c_3(y)$.

Viewing $\mathcal{ALC}$ from the modal logic perspective, atomic concepts can be interpreted simply as atomic *propositional* formulae. In this case the value restriction $\forall.$ becomes a modal operator since it is applied to formulae. Thus the above mentioned concept can be expressed by the propositional modal formula $\neg c_1 \vee \mathbf{K_r}(c_2 \wedge c_3)$. $\mathbf{K_r}(c_2 \wedge c_3)$ is to be read as "agent r *knows* proposition $c_2 \wedge c_3$," and means that in every world accessible for r, both c_2 and c_3 hold. Actually,

- the domain of an extension function can be read as a set of *worlds*.
- atomic concepts can be interpreted as the set of worlds in which they hold, if expounded as atomic formulae.

[1]This work was supported by the Commission of the European Communities and is part of Esprit Project AIMS (5210).

- atomic roles can be viewed as denoting *accessibility relations.*

Hence $d \in \mathcal{E}[\forall R.C]$ can be expounded as "in world d agent R knows proposition C." This illustrates that $\mathcal{ALC}$ is a notational variant of the propositional modal logic $\mathbf{K}_{(\mathbf{m})}$. For a brief introduction to $\mathbf{K}_{(\mathbf{m})}$ confer for example [2]. To demonstrate the utility of the correspondence, I exposed two of its immediate by-products in [8]. Namely, I gave an axiomatization of $\mathcal{ALC}$ and a simple proof that subsumption in $\mathcal{ALC}$ is PSPACE-complete, replacing the original six-page one in [9].

Terminological Logic and Dynamic Logic

Moreover, I have considered an extension of $\mathcal{ALC}$, called $\mathcal{TSL}$, comprising various role forming operators. In addition to the $\mathcal{ALC}$-operations, $\mathcal{TSL}$ contains both the identity role id and the composition $\circ$, the disjunction $\sqcup$, the transitive-reflexive closure *, the range restriction :, and the inverse $^{-1}$ of roles. Now it is important to realize that roles can be interpreted not only as accessibility relations but also as *nondeterministic programs.* In this case the domain of the extension function $\mathcal{E}$ is to be read as a set of program *states*, and $\langle d, e\rangle \in \mathcal{E}[R]$ denotes that there is an execution of the program R transforming state d into state e. Using this interpretation, compound terms can be expounded as follows:

- $d \in \mathcal{E}[\forall R.C]$ as "whenever program R terminates starting in state d, proposition C holds on termination"
- $R_1 \circ R_2$ as "run R_1 and R_2 consecutively"
- $R_1 \sqcup R_2$ as "nondeterministically do R_1 or R_2"
- R^* as "repeat program R a nondeterministically chosen number of times ≥ 0"
- R^{-1} as "run R in reverse"
- $\mathsf{id} : C$ as "proceed without changing the program state iff proposition C holds"

This illustrates that $\mathcal{TSL}$ is a notational variant of the propositional dynamic logic PDL with the *converse*-operator. Using this correspondence, one can easily prove that (a) it suffices to consider finite connected $\mathcal{TSL}$-models of exponential size, (b) $\mathcal{ALC}$ augmented with the transitive-reflexive closure of roles is EXPTIME-hard, and that (c) $\mathcal{TSL}$-subsumption can be computed in exponential time even w.r.t. a finite set of concept equations. Moreover, utilizing the correspondence one obtain an axiomatization of $\mathcal{TSL}$.

Since *features* (functional roles) correspond to *deterministic programs* in dynamic logic, it follows that adding them to $\mathcal{TSL}$ preserves decidability, although violates its Finite Model Property (FMP, for short). Surprisingly, adding both role-conjunction and features to $\mathcal{TSL}$ does not preserve decidability. All these results are summarized in Figure 1 and can be found in [8].

Name	Concept Operators	Role Operators	Notational Variant	Complexity of Subsumption	FMP
$\mathcal{ALC}$	concept names, $\sqcap, \neg, \forall., [\ \sqcup, \exists.\ ]$	role names, $[\ \sqcup, \circ, \mathsf{id}, :\ ]$	$\mathbf{K}_{(\mathbf{m})}$	PSPACE-complete [2, 9]	yes [2]
$\mathcal{ALC}_{reg}$	concept names, $\sqcap, \neg, \forall., [\ \sqcup, \exists.\ ]$	role names, $\sqcup, \circ, {}^*$	test-free PDL	EXPTIME-hard [1] EXPTIME-easy† [5]	yes [1]
$\mathcal{ALC}_{reg}+R:C$	concept names, $\sqcap, \neg, \forall., [\ \sqcup, \exists.\ ]$	role names, $\sqcup, \circ, {}^*, \mathsf{id}, :$	PDL	EXPTIME-hard [1] EXPTIME-easy† [5]	yes [1]
$\mathcal{ALC}_{reg}+R:C$ $+R\sqcap S$	concept names, $\sqcap, \neg, \forall., [\ \sqcup, \exists.\ ]$	role names, $\sqcup, \circ, {}^*, \mathsf{id}, :, \sqcap$	IPDL	EXPTIME-hard [1]	no [3]
$\mathcal{TSL}$	concept names, $\sqcap, \neg, \forall., [\ \sqcup, \exists.\ ]$	role names, $\sqcup, \circ, {}^*, \mathsf{id}, :, {}^{-1}$	CPDL	EXPTIME-hard [1] EXPTIME-easy† [3]	yes [1]
$\mathcal{FSL}$	concept names, $\sqcap, \neg, \forall., [\ \sqcup, \exists.\ ]$	[role names,] $\sqcup, \circ, {}^*, \mathsf{id}, :, {}^{-1}$, feature names	CDPDL	EXPTIME-hard [4] decidable [10]	no [3]
${}^{-1}$-free $\mathcal{FSLR}$	concept names, $\sqcap, \neg, \forall., [\ \sqcup, \exists.\ ]$	[role names,] $\sqcup, \circ, {}^*, \mathsf{id}, :, \sqcap$, feature names	IDPDL	undecidable [3]	no [3]

Term forming operators occurring in [ ] can be added without changing the expressive power or the computational complexity of the corresponding language.
†Even w.r.t. a finite set of concept equations and inequations [8].

Figure 1: Terminological Logics and their Notational Variants.

Further Issues

This work can be extended in two ways. First, one can further exploit the correspondences already established by carefully studying the corresponding theories of modal and dynamic logic. For example, I proved that subsumption in a syntactically restricted form of $\mathcal{TSL}$ with universal implications, known in dynamic logic as *partial completeness assertions*, is co-NP-complete. Secondly, one can establish additional correspondences. *Constants* in terminological logics, for instance, correspond to *names* (atomic formulae denoting single element sets) in dynamic logic. Similarly, temporal operators can easily be integrated into terminological logics. The reason is that temporal concepts such as $\mathsf{sometime}(TR, C)$ and $\mathsf{alltime}(TR, C)$ with TR being either **earlier** or **later** clearly correspond to the well-known modal operators of the *Tense Logic*.

References

[1] Michael J. Fischer and Richard E. Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Science*, 18:194–211, 1979.

[2] Joseph Y. Halpern and Yoram Moses. A Guide to the Modal Logics of Knowledge and Belief. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 480–490, Los Angeles, Cal., 1985.

[3] David Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 2, pages 497–604, Dordrecht, Holland, 1984. Reidel.

[4] Rohit Parikh. Propositional Dynamic Logics of Programms: A Survey. In E. Engeler, editor, *Proceedings of the Workshop on Logic of Programs*, volume 125 of *Lecture Notes in Computer Science*, pages 102–144, Berlin, FRG, 1979. Springer-Verlag.

[5] Vaughan R. Pratt. Models of Program Logics. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 115–122, San Juan, Puerto Rico, 1979.

[6] Klaus Schild. Undecidability of Subsumption in $\mathcal{U}$. KIT Report 67, Department of Computer Science, Technische Universität Berlin, Berlin, FRG, October 1988.

[7] Klaus Schild. Towards a Theory of Frames and Rules. KIT Report 76, Department of Computer Science, Technische Universität Berlin, Berlin, FRG, December 1989.

[8] Klaus Schild. A Correspondence Theory for Terminological Logics: Preliminary Report. In *Proceedings of the IJCAI'91*, 1991. To appear.

[9] Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[10] Moshe Y. Vardi. The Taming of Converse: Reasoning about Two-Way Computations. In R. Parikh, editor, *Proceedings of the Workshop on Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 413–424, Berlin, FRG, 1985. Springer-Verlag.

Integrating Time into Terminological Logics

Albrecht Schmiedel
Deutsches Herzzentrum Berlin
atms@cs.tu-berlin.de

Presumably there are many ways of integrating time into terminological logics. To begin with, I will point out a more or less shallow mode of integration which seems to be useful in many cases all the same. For a deep integration a number of choices concerning the semantic model and various degrees of expressivity have to be made. I will argue for the potential usefulness of a fairly expressive approach such as [6] in a monitoring scenario.

A shallow integration of time

Rather than building time right into the semantic structure of a terminological logic and therefore having to deal with time everywhere, a more limited approach stays with the static, timeless model and gives time no special status. Instead, time is introduced as a particular *concrete domain* in the sense of [1]:

- time intervals are taken as the individuals of the concrete domain[1],
- constraints[2] on the duration, absolute bounds, and granularity of intervals as unary predicates, and
- Allen's interval relations as binary predicates structuring the domain.

This is an *admissible* concrete domain, since the predicates provided are closed under negation, and satisfiability for conjunctions of these predicates is assumably decidable.

Adopting this approach allows the user to describe (abstract) objects in terms of features taking time intervals as values. Concepts can be formulated in terms of temporal constraints on those features. For example, we might model 'birthdays' using time intervals constrained to granularity and duration 'day'. We can then define 'a person whose birthday is before 1950', or 'a person whose birthday is after that of her mother'. This is good enough for cases where it is sufficient to model time as one property of objects among others, and express relationships between objects in terms of the corresponding

[1]Note that the approach of [1] for representing time intervals is different: they use the real numbers structured by $<, \leq, >, \geq, =, \neq$ as the concrete domain and define intervals and interval relations on top of it.

[2]cf. [5] for this type of constraint language

relationships of the (temporal) property. This is definitely a useful combination of the general-purpose terminological reasoner and the specialized temporal constraint reasoner built into the concrete domain.

But things get more difficult when we need to explicitly represent changing properties or relations of an object over time. Of course this could be done up to a point by introducing auxiliary objects; but this would be ad-hoc, and generally adopting this approach would jeopardize the goal of providing high-level, easy-to-understand, and intuitive knowledge representation primitives.

What other plausible alternative routes to take are there for integrating time into terminological logics? Poesio [3] pursues a kind of 'reification' of role-relationships, thus turning such relationships into objects of their own right, and therefore being able to add descriptions to such objects, e.g. time of validity. This is certainly an interesting approach (reminding one of Hobbs' 'ontological promiscuity' [2]), but it would have to be worked out more formally to make an adequate asessment possible. One of the main difficulties with this approach would be (as far as I can tell) to achieve a seamless integration with non-temporal terminological logic. One would have to refer to reified states from within terms in order to be able to define concepts with temporal structure.

Deep integration of time

In the following I will discuss some issues involved in a *deep* integration of time. I will call an integration of time deep when it is built right into the semantic structure, i.e., terms of a temporal terminological language are interpreted in a structure which explicitly mentions a time domain besides the domain of individuals. In general, this means that the extensions of concepts and roles will be time-dependent. This opens up completely new representational possibilities compared to the static model above:

- individuals can have different properties at different times and can participate in varying relations with other individuals time-dependently, and
- concepts can describe individuals in terms of their patterns of variation, thus enabling a kind of *temporal abstraction* in addition to the structural abstraction in the non-temporal case.

Various degrees of expressivity can be envisaged when designing a temporal terminological logic.

Firstly, should the time domain be point-based or interval-based? Interval-based approaches are more expressive, since they might include the point-based ones (zero-length intervals), and they can potentially treat the different temporal types, such as downward-hereditary, concatenable, solid, etc. propositions (cf. [7]). The latter is interesting when integrating various types of temporal knowledge from different levels of abstraction is an issue (see the section on applications below). In my approach I have taken this route [6].

Secondly, what kind of term-forming operators should be included in the temporal terminological logic? Should temporal variables (together with temporal constraints) be allowed inside temporal term-forming operators? Or should they remain implicit? Clearly, term-forming operators such as '(sometime-earlier *concept*)', '(since *concept*)' as proposed by [4] refer to an implicit temporal variable and a temporal relation, which is in the spirit of the variable-free syntax of non-temporal term-forming operators. Also, their meaning is intuitively clear. On the other hand, certain temporal patterns cannot be expressed this way, and nested temporal operators of this kind can be difficult to understand.

A possible area of application: monitoring

It is important to examine carefully possible areas of application for temporal terminological logics, because although it is easy to make out a historical or temporal dimension in almost every domain, explicitly representing time does complicate matters considerably. Much has to be made explicit what is naturally hidden in a 'time-less' knowledge base. For example, update time and time of validity have to be differentiated and explicitly dealt with in a deep temporal model; this isn't important when the knowledge base is meant to model only one current state of affairs. Often enough, a shallow integration by means of a concrete domain as indicated above will be all one wants.

But in some areas change and patterns of change are the actually interesting aspects. One area (among others such as planning) are computer systems monitoring processes that produce large amounts of time-dependent data such as environmental surveillance, production plant monitoring, or intensive care. Integration and combination of data, detecting relevant types of events, and data abstraction and reduction are the main tasks for this type of application.

Here, a sufficiently expressive temporal terminological logic could provide a *human window* to the mass data generated by such a process. The services provided by a temporal terminological logic could be:

- Integrate primitive data (from sensors etc.) as well as all kinds of derived information (statistical abstractions such as averages, standard deviations, etc., qualitative abstractions, and derived events types) within one unified formalism.
- Define all interesting and relevant states, events, statistics in terms of (structural and temporal) abstractions from primitive data (and each other).
- Define triggers for actions in terms of (temporal) concepts.
- Perform data reduction: from a process generating large amounts of primitive data, keep only a small amount of interesting information describing the history of the process using defined concepts and relations. (*data* vs. *structured knowledge* about the *running history*.)
- Browse the running history from various perspectives:
  - What happened in a certain time interval?
  - How did a certain object change in the course of time?
  - What is the history of certain event types?
- Generate explanations for globally defined events, states, and measures exploring how their definitions were instantiated.

Since recognition on the basis of completely given ground data would be the main mode of operation in this type of application (rather than consistency checking) completeness of the subsumption checker is not of primary concern. On the other hand, an interval-based approach seems essential; otherwise, integration of concepts on successive levels of granularity would be difficult. Further, complex temporal patterns should be easily expressible, including constraints on duration and absolute times.

References

[1] F. Baader, P. Hanschke, A Scheme for Integrating Concrete Domains into Concept Languages, Research Report RR-91-10, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, 1991

[2] J. R. Hobbs, "Ontological Promiscuity," *Proc. 23rd Annual Meeting ACL*, Chicago, Ill., 1985, 61–69.

[3] M. Poesio, Towards a Hybrid Representation of Time, *Proc. ECAI-88*, Munich, 1988.

[4] K. Schild, Integration of Terminological Logic and Tense Logic, unpublished draft, Berlin, 1991

[5] A. Schmiedel, Temporal Constraint Networks, KIT Report 69, Fachbereich Informatik, Technical University Berlin, November 1988.

[6] A. Schmiedel, A Temporal Terminological Logic, *Proc. AAAI-90*, Boston (Mass.), 640–645, 1990.

[7] Y. Shoham, Temporal Logics in AI: Semantical and Ontological Considerations, *Artificial Intelligence* 33(1), 89–104, 1987.

Sorted Feature Terms and Relational Dependencies

Roland Seiffert
IBM Deutschland GmbH
Institute for Knowledge Based Systems
P.O. Box 80 08 80
D-7000 Stuttgart 80
Fed. Rep. of Germany
e-mail: seiffert@ds0lilog.bitnet

1 Overview

Our current research focusses on the development of a unification-based grammar formalism mainly for the use within the LILOG project at IBM Germany for building a HPSG-style grammar of German. STUF integrates *feature terms* with *sorts* and *recursive definitions of relations*. This is in the spirit of a new consensus of recent formalisms used in computational linguistics that can be characterized as *principle-based* approaches to grammar. Principles state relations over (typed) feature structures, incorporating phonological strings and constituent structures as integral parts. Examples for such formalisms are HPSG ([PS87]), TFS ([ZE90]), CUF ([DE91]), and also the knowledge representation language LIFE ([AKP90]). We show how the concepts of these formalisms can be further enriched and integrated into one formalism, STUF. A rigorous though simple semantics for the complete formalism is given in [DS91]. It is argued that exactly this chosen combination allows for a very elegant formulation of HPSG-style grammars. Also, we give a translation of relational dependencies to definite clauses of first-order logic that fits exactly into the generalized Constraint Logic Programming scheme of [HS88]. This opens up the treasure of results and techniques known in CLP and promises that an efficient implementation of STUF is possible. A detailed description of the formalism is given in [DS91].

2 The Formalism

Central to our formalism is the notion of a feature term coined by [KR86] and extended and generalized by [Smo88, Smo89]. It allows us to specify sets of feature structures in a linearized feature-matrix oriented notation. This differentiation between feature structures and their descriptions is crucial since it allows us to extend the descriptional devices to include e.g. disjunction or negation without having to stipulate new kinds of underlying structures. We even can abstract away from the concrete notion of a feature structure. We only assume a domain of discourse to contain unspecified elements, where feature applications are reflected by functional dependencies between those elements. In the following the letters $s, t, t_1, \ldots$ will always denote feature terms. The syntactic forms of feature terms are given by the context-free production in Figure 1.

s, t	$\longrightarrow$	x	a variable
	$\mid$	A	a sort
	$\mid$	$f{:}t$	feature selection
	$\mid$	$s\&t$	conjunction (intersection)
	$\mid$	$s;t$	disjunction (union)
	$\mid$	$\neg t$	negation (complement)

Figure 1: The Syntax of Feature Terms

The simplest forms are variables and sorts. Feature terms may contain variables to state sharing of structures, i.e. they serve the same purpose as path equations in Kasper-Rounds logic. Sorts are described in more detail below. A term $f : t$ denotes the set of those elements for which the feature f leads to an element in the denotation of t. Conjunction, disjunction, and negation are set intersection, union, and complement, respectively.

2.1 Sorts

The intended meaning of sorts is to denote subsets of the domain of discourse. The integration of sorts into feature terms follows [Smo88]. The main difference is, that we made it possible to define the sort structure within our formalism.

Sorts come in three varieties: *atoms*, *primitive sorts*, and *defined sorts*. For atoms we use the letters a, b, c, ... and for sorts other than atoms we write A, B, C, ... An *atom* is assumed to denote a singleton subset of the domain that is disjoint to the set denoted by any other atom or primitive sort. A *primitive sort* denotes an arbitrary, unspecified subset of the domain. *Defined sorts* can be built from arbitrary sorts using boolean connectives in a definition of the form $A = sexpr$. Figure 2 shows the syntax of these sort expressions. The connectives are interpreted in the usual way, i.e. for example $A \sqcap B$ denotes the intersection of the sets denoted by A and B.

$sexpr \longrightarrow$	A
	a
	$sexpr \sqcap sexpr$
	$sexpr \sqcup sexpr$
	$\neg sexpr$

Figure 2: The Syntax of Sort Expressions

Obviously, the sets denoted by sort expression defined in this way form a distributive lattice with set inclusion being the order relation and intersection and union being the meet and join operations. In our system these operations are implemented using a propositional theorem prover that operates on bit vectors for the internal representation of sorts[1]. The description of that algorithm lies beyond the scope of this paper.

Sorts serve two purposes, a syntactic and a semantic. In the syntax they are used by a typing scheme similar to that of [Car90]. For example, for well-typed expressions for which a certain sort is specified (or inferred) only certain features are allowed, whose values in turn have to obey sort restrictions. However, in this paper we will not give details of this syntactic use of sorts.

Semantically sorts are a means to coarsely structure the domain of interest. The sorts definable in our system can constitute a hierarchy, and one way of refining the information about a certain object which is known to be of some sort A would be to go to a subsort of A. Compared to the usual atomic values in feature structures, the unification of two different sorts does not necessarily lead to inconsistency, but instead depends on the sort hierarchy and we get the greatest common subsort of the two, if it exists. Also, sorts are compatible with feature specifications, i.e. objects denoted by sorts may have features.

Since sorts can be handled very efficiently using a specialized propositional theorem prover we expect that they will be employed very much to substitute for a lot of disjunctions that without sorts would be "structural", i.e. they would have to be treated as really disjunctive feature terms. Dealing with those is a well-known source for very hard computational problems.

2.2 Integrating Relations

We extend the syntax of feature terms to include the form:

$A(t_1, \ldots, t_n)$ a relational dependency

where t_i are all feature terms. Hence, relational dependencies are used in our syntax as function applications in disguise. The meaning of such a term depends on the n+1-place relation A, which can be introduced through relation definitions, as described below. A relational dependency term $A(t_1, \ldots, t_n)$ now denotes the set of values which the additional argument, let's call it the 0-argument, can take, when the other arguments are in the denotations of their respective t_i. Suppose for example a 3-place relation *append* on lists whose 0-argument is the concatenation of the other two arguments. The definition of this relation is given below. Now, the term with the variables X, Y, and Z

```
f: X  &
g: Y  &
h: Z  &
i: append(X, append(Y,Z))
```

denotes a structure whose value of the feature i is the concatenation of the values of the features f, g, and h. The meaning of relational dependencies does not imply an order of evaluation as one might assume for this function application syntax. Our semantics is

[1]similar to that of [She89]

completely declarative, also in this respect. Hence, a relational dependency may be used to generate its arguments from its 'result' value, or to propagate side-effects from one argument to another.

For example, conjoining the above term with

```
g: [b] &
i: [a,b,c]
```

would only yield a non-empty denotation if the values of f and h are taken to be $[a]$ and $[c]$, respectively.[2]

A n+1-place relation A is defined through a set of defining clauses of the form:

$$A(t_1, \ldots, t_n) := t_0. \qquad \text{where all } t_i \text{ are feature terms}$$

Multiple defining clauses for one relation are taken disjunctively. The meaning of such a defining clause is that the terms t_0 to t_n give us a sufficient condition on the description of objects u_0 to u_n, respectively, in order to be in the relation A. For example, the definition of the recursive relation *append* can be given as follows.

```
append([],L) := L.
append([F|R],L) := [F|append(R,L)].
```

Actually, this means that we are proposing some sort of logical programming language where a very powerful term syntax is used, including disjunction and relational dependencies, making the relational body of a clause superfluous. Notice that the right-hand side of a defining clause is just the term for the implicit 0-argument.

An important observation is that unary relations and sorts are semantically the same, at least in effect. Also syntactically there is no difference between "application" of a unary relation and a sort. Since the only difference between a unary relation and an ordinary sort is that a unary relation is defined via general feature terms, we will call such relations simply generally-defined sorts. The term "relational dependencies" shall henceforth only refer to terms of at least one parameter, i.e. terms that refer to relations of at least two arguments.

In [DS91] you can find a discussion on the use of relational dependencies. The two most important arguments are: First, you can arrive at more concise and clear grammars makeing use of parameters. Second, explicit knowledge of "result values" makes data structures smaller and allows for garbage collection.

3 Implementational Issues

In this section we will present some of the basic ideas underlying our current STUF implementation. First, recall that our grammars employ a quasi-functional notation to define relations. We can make these relations explicit by systematically introducing an additional argument for all relations. We now present a translation function *trans* (Figure 4) that converts a given definition $r(\vec{s}) := t$ into a new, equivalent definition of the form $r(X, \vec{X}) \leftarrow t'$, where t' is a formula built of conjunctions and disjunctions of *basic constraints* (Figure 3).

$f(X) = Y$, where f is a feature, X and Y are variables
$X \in A$, where A is a sort, X is a variable
$X = Y$, where X and Y are variables
$X \neq Y$, where X and Y are variables
$r(X_0, \ldots, X_n)$, where r is a (defined) n-ary relation; $X_0, \ldots, X_n$ is usually written as $\vec{X}$

Figure 3: The Basic Constraints

trans($r(\vec{s}) := t$)	$\Rightarrow$	$r(X, \vec{X}) \leftarrow$ trans(X,t) $\wedge$ trans(X_i,s_i), for all X_i in $\vec{X}$ and corresponding s_i in $\vec{s}$, and X and all X_i are new variables
trans(X,$f : t$)	$\Rightarrow$	$f(X) = Y \wedge$ trans(Y,t), where Y is a new variable
trans(X,A)	$\Rightarrow$	$X \in A$
trans(X,$\neg A$)	$\Rightarrow$	$X \in \bar{A}$
trans(X,Y)	$\Rightarrow$	$X = Y$
trans(X,$\neg Y$)	$\Rightarrow$	$X \neq Y$
trans(X,$t_1 \& t_2$)	$\Rightarrow$	trans(X,t_1) $\wedge$ trans(X,t_2)
trans(X,$t_1 ; t_2$)	$\Rightarrow$	trans(X,t_1) $\vee$ trans(X,t_2)
trans(X,$r(\vec{t})$)	$\Rightarrow$	$r(X, \vec{X}) \wedge$ trans(X_i,t_i), for all X_i in $\vec{X}$ and corresponding t_i in $\vec{t}$, and all X_i are new variables

Figure 4: The Translation Function *trans*

Note that we don't have a translation scheme for general negation $\neg t$. This is not an accident as will become clear soon.

As an example consider the definitions for `subcat_principle` and `append` from our HPSG fragment:

```
subcat_principle(syn: loc: subcat: append(S1,S2),S2) :=
        syn: loc: subcat: S1.

append(nil,Y) := Y.
append(cons&(first:X1)&(rest:X),Y) :=
        cons&(first:X1)&(rest:append(X,Y)).
```

The function *trans* applied to all definitions in the example yields:

[2]or anything that is subsumed by that

```
subcat_principle(M, HD, CDs) ←
    syn(M) = M1 ∧ loc(M1) = M2 ∧ subcat(M2) = S1∧
    syn(HD) = HD1 ∧ loc(HD1) = HD2 ∧ subcat(HD2) = S∧
    append(S, S1, CDs).

append(Y, X, Y) ←
    X ∈ nil.
append(Z, X, Y) ←
    Z ∈ cons ∧ first(Z) = X1 ∧ rest(Z) = Zs∧
    X ∈ cons ∧ first(X) = X1 ∧ rest(X) = Xs∧
    append(Zs, Xs, Y).
```

If we exclude general negation then each of the definitions resulting from *trans* can be easily transformed into an equivalent set of *normal form clauses*. The normal form is defined as

$$r_0(\vec{X_0}) \leftarrow \phi \wedge r_1(\vec{X_1}) \wedge \ldots \wedge r_n(\vec{X_n})$$

ϕ is a formula containing arbitrary conjunctions and disjunctions of basic constraints except relational atoms. If our system contains only definitions in normal form—definite clauses—then this fits nicely into the refined Constraint Logic Programming scheme described in [HS88]. In fact, our definite relations correspond exactly to the relational extensions of simple feature logic as described in [Smo88] for which efficient constraint solvers exist. This immediately gives us an operational semantics for solving the relational constraints, which is a generalization of SLD-resolution. Our implementation of STUF is based on this SLD-resolution scheme and thanks to [HS88] we know that this approach is sound and complete.

Another observation is that a very common optimization technique from conventional logic programming can be integrated into our framework: Some of the (non-relational) constraints of ϕ are associated with the head of a definite clause (ϕ_{Head}) and others with the body (ϕ_{Body}), i.e. $\phi = \phi_{Head} \wedge \phi_{Body}$.

$$r_0(\vec{X_0}) \leftarrow \phi_{Head} \wedge \phi_{Body} \wedge r_1(\vec{X_1}) \wedge \ldots \wedge r_n(\vec{X_n})$$

ϕ_{Head} should be very simple but impose very strong constraints. Then it can be used to efficiently cut down the search space significantly. When selecting a clause, we first unify with ϕ_{Head} and if this fails, we reject the clause immediately. Typical constraints in ϕ_{Head} are sort restrictions on the variables occurring in the head of the clause.

4 Current Work

Most of the work currently being done for STUF deals with devising proof strategies that allow for an efficient processing of grammars written in STUF both for the analysis of sentences and for generation. As a starting point we consider very general techniques, e.g. partial execution of programs at compile time, detecting and preferring of deterministic "subproofs" at run time, ... We also try to learn from the experience we have in context-free based parsing and to incorporate similar strategies and heuristics into our new approach.

References

[AKP90] Hassan Aït-Kaci and Andreas Podelski. Is there a meaning to LIFE? Draft paper, Nov. 1990.

[Car90] Bob Carpenter. Typed feature structures: Inheritance, (in)equality and extensionality. In *Proceedings of the Workshop on Inheritance in Natural Language Processing*, Tilburg University, The Netherlands, 1990.

[DE91] Jochen Dörre and Andreas Eisele. A comprehensive unification-based grammar formalism. Deliverable R3.1.B, DYANA — ESPRIT Basic Research Action BR3175, 1991. to appear.

[DR90] Jochen Dörre and William C. Rounds. On Subsumption and Semi-Unification in Feature Algebras. In *Proceedings of the 5th Annual Symposium on Logic in Computer Science*, pages 300–310, Philadelphia, PA., 1990.

[DS91] Jochen Dörre and Roland Seiffert. Sorted Feature Terms and Relational Dpendencies. IWBS Report 153, IBM Deutschland GmbH, Institute for Knowledge Based Systems, 1991.

[HS88] Markus Höhfeld and Gert Smolka. Definite relations over constraint languages. LILOG Report 53, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, W. Germany, October 1988. To appear in the Journal of Logic Programming.

[KR86] Robert T. Kasper and William C. Rounds. A logical semantics for feature structures. In *Proceedings of the 24th Annual Meeting of the ACL, Columbia University*, pages 257–265, New York, N.Y., 1986.

[PS87] Carl Pollard and Ivan A. Sag. *Information-Based Syntax and Semantics.* CSLI Lecture Notes 13. Center for the Study of Language and Information, Stanford University, 1987.

[Rea89] Mike Reape. A logical treatment of semi-free word order and bounded discontinuous constituency. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, pages 103–110, Manchester, England, 1989.

[She89] M. J. Shensa. A computational structure for the propositional calculus. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan, USA, 1989.

[Smo88] Gert Smolka. A feature logic with subsorts. LILOG Report 33, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, W. Germany, May 1988. To appear in the Journal of Automated Reasoning.

[Smo89] Gert Smolka. Feature constraint logics for unification grammars. IWBS Report 93, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, W.

Germany, November 1989. To appear in the Proceedings of the Workshop on Unification Formalisms—Syntax, Semantics and Implementation, Titisee, The MIT Press, 1990.

[ZE90] Rémi Zajac and Martin Emele. Typed unification grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, Finland, 1990.

Using Terminological Logics in a Problem Solver

William Swartout
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
USA

The goals of the Explainable Expert Systems project (EES) have been to construct an framework for building expert systems that:

- enhances their explanation capabilities [MS89],
- eases their maintenance and extension [NSM85], and
- allows system builders to rapidly construct prototypes [SNPS89].

In the references cited above we have argued that conventional expert system frameworks are seriously limited in providing these capabilities. These limitations stem in part from problems in their underlying knowledge representation—specifically the use of low-level rules that implicitly encode and compile together different kinds of knowledge. Because different kinds of knowledge are not distinguished, this implicit, intertwined representation of knowledge makes a system less modular and understandable and hence more difficult to modify or explain [Cla83, Swa83]. Because knowledge is intertwined and multiple concerns may be expressed in a single rule, it is also more difficult to re-use knowledge across systems, hence making it more difficult to use knowledge from an existing system to construct a rapid prototype of a new system.

We deal with these problems by taking a different approach to the construction of expert systems. We begin by representing knowledge at a more abstract level and providing representations that allow us to explicitly distinguish and separate different kinds of knowledge. The kinds of knowledge that we distinguish include:

- terminology, which defines the terms in the domain and provides the 'building blocks' out of which the rest of the knowledge base is constructed,
- a domain model, which describes how the domain 'works' (e.g. a causal model or a circuit schematic), and
- problem solving knowledge, expressed as a set of plans, which tells the system how to perform tasks such as diagnosing a network or assessing the state of a patient.

The EES framework (rather than the system builder) takes responsibility for linking together these various kinds of knowledge to actually solve problems. This approach allows a system builder to work at a more abstract and explicit level of representation, and increases the modularity of the overall system, thereby helping us achieve our goals of enhanced explanation, easier maintenance, and support for rapid prototyping.

The linkage of different kinds of knowledge is a critical step in our approach, and it depends on our use of a term subsumption-based knowledge representation (Loom). The domain model is constructed using the conceptual structures and assertional capabilities of Loom. Representing problem-solving knowledge is more of a challenge. Goals to be achieved by the problem solver are represented as conceptualized verb clauses based on a case grammar approach [Fil68]. Thus, the goal of 'put block-a on the table' would be represented as a specialization of the verb 'put' with slots filled to specify the object to be manipulated (block-a) and its destination (table). *Capability descriptions* are associated with plans, and describe what the plans can do. Capability descriptions are also represented using conceptual structures, but special mechanisms have to be provided to allow variables to appear in capability descriptions so that parameters can be passed from goals into plans. We use the classifier and realization mechanisms of Loom as a sort of pattern-matcher to find plans that are capable of achieving goals.

One advantage of this approach to representing goals and plans is that it gives a goal an independent meaning which is based on the conceptual structur from which it is composed, unlike conventional systems where a goal acquires its meaning based solely on how plans in the system react to it. An additional advantage of the approach is that it allows us to define domain-independent techniques for reformulating a posted goal into other goals when no plans can be found for achieving the original goal. These reformulations are based on the meaning of the goal itself. The system makes use of facts about the domain expressed in the domain model to perform the reformulation, and it is through reformulations that much of the domain knowledge becomes integrated into the problem solving process (see [NSM85] for a more complete description).

Generalizing from the specific concerns of EES, there are some observations we can make. Traditionally, term subsumption knowledge bases have been regarded as repositories for knowledge that perform certain kinds of deductive inference with reasonable alacrity. It seems that th view may be too limiting. While many problems can be formulated as deductive problems, there are many problems that do not fit naturally within a deductive framework. Furthermore, our work in EES, and of others [YNM89, Yen90], suggests that there may be a lot to be gained by integrating terminological reasoners with other kinds of problem solving architectures such as planners or rule based systems. As one of the topics for the workshop, I would like to suggest that we consider how terminological representation systems can be integrated with other problem solving architectures. Some specific questions we might consider include:

- How does the integration of a problem-solver with a terminological KB affect expressivity needs?
- What sorts of software interfaces need to be provided so that different problem solving systems can be interfaced? What approaches have worked? What haven't?
- Integrating a problem solver raises the question of how much of the system's overall processing should be done by the reasoning mechanisms provided by the knowledge

representation and how much should be done by the problem solver. What are the tradeoffs?

Acknowledgements

The research described in this paper was supported by the Defense Advanced Research Projects Agency under Grant MDA 903-81-C-0335 and under a NASA Ames cooperative agreement number NCC 2-520.

References

[Cla83] W. Clancey. The epistemology of a rule-based expert system: A framework for explanation. *Artificial Intelligence*, 20(3):215–251, 1983.

[Fil68] C. Fillmore. The case for case. In *Universals in Linguistic Theory.* Holt, Rinehart and Winston, 1968.

[MS89] Johanna D. Moore and William R. Swartout. A reactive approach to explanation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 20-25 1989. IJCAI.

[NSM85] R. Neches, W. R. Swartout, and J. D. Moore. Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Transactions on Software Engineering*, SE-11(11):1337–1351, November 1985.

[SNPS89] W. Swartout, H. Nordin, C. Paris, and S. Smoliar. Toward a rapid prototyping environment for expert systems. In *Proceedings of the 13th German Workshop on Artificial Intelligence*, pages 438–454, 1989.

[Swa83] W. Swartout. Xplain: A system for creating and explaining expert consulting systems. *Artificial Intelligence*, 21(3):285–325, September 1983. Also available as ISI/RS-83-4.

[Yen90] J. Yen. A principled approach to reasoning about the specificity of rules. In *Proceedings of AAAI-90, the Eighth National Conference on Artificial Intelligence*, 1990.

[YNM89] J. Yen, R. Neches, and R. MacGregor. Using terminological models to enhance the rule-based paradigm. In *Proceedings of the Second International Symposium on Artificial Intelligence*, 1989.

Terminological Knowledge Representation: A Proposal for a Terminological Logic

Franz Baader, Hans-Jürgen Bürckert, Jochen Heinsohn, Bernhard Hollunder, Jürgen Müller, Bernhard Nebel, Werner Nutt, Hans-Jürgen Profitlich
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Postfach 2080, W-6750 Kaiserslautern
Stuhlsatzenhausweg 3, W-6600 Saarbrücken 11
Germany

Abstract
This paper contains a proposal for a terminological logic. The formalisms for representing knowledge as well as the needed inferences are described.

1 Introduction

An important aspect of intelligence is the use of existing knowledge. In order to realize this in AI-Systems we need both adequate methods to represent knowledge and effective procedures to retrieve and reuse the needed knowledge. One of the basic mechanisms of human knowledge representation and processing is the division of the world into classes or concepts ("find the right pigeonhole") which usually are given with a hierarchical structure.

Let us consider some knowledge base about families and relationships. We have to deal with persons which are of sex male or female. We have parents, mothers, fathers etc. A verbal description of this knowledge might be as follows:

- *Persons* are of *sex Male* or *Female.*
- *Woman* is a *Person* with *sex Female.*
- *Man* is a *Person* with *sex Male.*
- *Parents* are defined as *Persons* which have some *child* (which is also a *Person*).
- *Mothers* are defined to be *Parents* with *sex Female.*
- *Fathers* are defined to be *Parents* with *sex Male.*
- *Mother_with_many_children* is defined as *Mother* with at least three *children.*

We also have individuals (or objects) which are instances of concepts. For example,

- *John* is a *Father.*
- *Tom* is a *child* of *John.*
- *Mary* is a *Woman.*

Now every knowledge representation system should offer a couple of services that allow to arrange, manage, modify or retrieve information of the above kind. It should be able to answer the following questions:

- Is an introduced concept defined in a meaningful way at all (or does it denote the empty concept in all worlds) ? (*satisfiability*)
- Is a concept more general than another one ? (*subsumption*)
- Where exactly is the concept situated in a concept hierarchy ? (*classification*)
- Is the represented knowledge consistent ? (*consistency*)
- What facts are deducible from the knowledge ? (*instantiation*)
- Which are the concepts an object is instance of ? (*realization*)
- Which are the instances of a given concept ? (*retrieval*)

Building such a system we are confronted with the following questions:

1. How can the above properties been found out at all ?

And then, if we know procedures that might do this:

2. How can we find out, whether the procedures really do what they should do ?
3. How efficient are these procedures ?
4. How efficient may an optimal procedure for the problem be ?

Terminological logics based on concept description languages like KL-ONE [BS85] are such formalisms that make classification, description of relations among the classes and especially their hierarchical structure possible. However, concept description languages are not only one among a lot of possibilities, but meanwhile they offer compared to other KR-formalisms some fundamental advantages:

- There is a well understood *declarative* semantics.
  This means that the meaning of the constructs is not given operationally, e.g. by the implementation ("John is a father", because my system answers to the question "What is John?" just "father"), but the meaning is given by its description and its models ("John is a father", because he is a father in all models—in all worlds—where the description suits to.)
- There is a characterization of the tasks of the KR-systems by the declarative semantics.

- There is a number of procedures and algorithms that realize these tasks, and their properties are well investigated now. Important properties are

  1. Correctness
     (If the system answers "John is a father", then John is a father within the meaning of the semantics—that is in all suitable worlds.)
  2. Completeness
     (The system answers "John is a father", if John is a father within the meaning of the semantics.)
  3. Decidability, Complexity
     (Are the services decidable at all, and how fast are they executable ?)

If we want to design a knowledge base, we first need a formal language that we can use. In the following we will present a proposal for a terminological language in both abstract form and machine readable form (LISP notation). As a kernel, our language contains all the constructs provided by $\mathcal{ALC}$ [SS88] and some additional operators which (sometimes?) can be translated into $\mathcal{ALCFNR}$ [HN90].

2 Symbols

The terminological language is based on the following primitives, the symbols of the alphabet:

- Concept names: *CN*
- Role names: *RN*
- Attribute names: *AN*
- Individual names: *IN*
- Object names: *ON*

Examples with respect to our introductory example are: `Person`, `Woman`, `Man`, `Parent` are concept names, `child` is a role name, `sex` is an attribute name, `Male` and `Female` are individual names, and `John` and `Mary` are objects names.

With this primitives we are allowed to form more complex expressions as specified in the next two sections:

- Concept expressions: C
- Role expressions: R
- Attribute expressions: A

The meaning of these is given by interpretations $\mathcal{I}$. They consist of a set $\Delta^{\mathcal{I}}$—the domain—and an interpretation function $\cdot^{\mathcal{I}}$, that assigns a set

$$CN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$$

to each concept name CN, a set-valued function (or equivalently a binary relation)

$$RN^{\mathcal{I}} : \Delta^{\mathcal{I}} \longrightarrow 2^{\Delta^{\mathcal{I}}} \qquad (RN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$$

to each role name RN, a single-valued partial function

$$AN^{\mathcal{I}} : dom\, AN^{\mathcal{I}} \longrightarrow \Delta^{\mathcal{I}},$$

where $dom\, AN^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to each attribute name AN, and an element

$$I^{\mathcal{I}} \in \Delta^{\mathcal{I}}$$

to each individual name IN and object name ON. We assume that different individuals and objects denote different elements in every interpretation. This property is called *unique name assumption* and is usually assumed in the database world.

3 Concept Forming Operators

Besides the concept, role, and attribute names our alphabet includes a number of operators that permit to compose more complex concepts, roles, and attributes. We allow for the following concept forming operators:

Concrete Form	**Abstract Form**	**Semantics**
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
(and $C_1 \ldots C_n$)	$C_1 \sqcap \ldots \sqcap C_n$	$C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
(or $C_1 \ldots C_n$)	$C_1 \sqcup \ldots \sqcup C_n$	$C_1^{\mathcal{I}} \cup \ldots \cup C_n^{\mathcal{I}}$
(not C)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
(all R C)	$\forall R : C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$
(some R)	$\exists R$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \neq \emptyset\}$
(some R C)	$\exists R : C$	$\{d \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}$
(atleast n R)	$\geq nR$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert \geq n\}$
(atmost n R)	$\leq nR$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert \leq n\}$
(exact n R)	nR	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d)\rvert = n\}$
(atleast n R C)	$\geq nR : C$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert \geq n\}$
(atmost n R C)	$\leq nR : C$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert \leq n\}$
(exact n R C)	$nR : C$	$\{d \in \Delta^{\mathcal{I}} \mid \lvert R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}\rvert = n\}$
(eq R_1 R_2)	$R_1 = R_2$	$\{d \in \Delta^{\mathcal{I}} \mid R_1^{\mathcal{I}}(d) = R_2^{\mathcal{I}}(d)\}$
(neq R_1 R_2)	$R_1 \neq R_2$	$\{d \in \Delta^{\mathcal{I}} \mid R_1^{\mathcal{I}}(d) \neq R_2^{\mathcal{I}}(d)\}$
(subset R_1 R_2)	$R_1 \subseteq R_2$	$\{d \in \Delta^{\mathcal{I}} \mid R_1^{\mathcal{I}}(d) \subseteq R_2^{\mathcal{I}}(d)\}$
(in A C)	$A : C$	$\{d \in dom\, A^{\mathcal{I}} \mid A^{\mathcal{I}}(d) \in C^{\mathcal{I}}\}$
(is A IN)	$A : IN$	$\{d \in dom\, A^{\mathcal{I}} \mid A^{\mathcal{I}}(d) = IN^{\mathcal{I}}\}$
(eq A_1 A_2)	$A_1 = A_2$	$\{d \in \Delta^{\mathcal{I}} \mid A_1^{\mathcal{I}}(d) = A_2^{\mathcal{I}}(d)\}$
(neq A_1 A_2)	$A_1 \neq A_2$	$\{d \in \Delta^{\mathcal{I}} \mid A_1^{\mathcal{I}}(d) \neq A_2^{\mathcal{I}}(d)\}$
(subset A_1 A_2)	$A_1 \subseteq A_2$	$\{d \in \Delta^{\mathcal{I}} \mid d \in dom\, A_1^{\mathcal{I}} \Rightarrow d \in dom\, A_2^{\mathcal{I}} \wedge A_1^{\mathcal{I}}(d) = A_2^{\mathcal{I}}(d)\}$
(oneof $IN_1 \ldots IN_n$)	$\{IN_1, \ldots, IN_n\}$	$\{IN_1^{\mathcal{I}}, \ldots, IN_n^{\mathcal{I}}\}$

Examples: The concept *mother* can be described as

$$\texttt{Person} \sqcap (\texttt{sex} : \texttt{Female});$$

Mother_with_many_children can be described as

$$\texttt{Mother} \sqcap (\geq 3\texttt{child} : \texttt{Person});$$

Father_with_sons_only can be described as

$$\texttt{Parent} \sqcap (\texttt{sex} : \texttt{Male}) \sqcap (\texttt{child} = \texttt{son}).$$

Please note that the semantics of $A_1 = A_2$ and $A_1 \neq A_2$ for attributes is defined analogously to the semantics of $R_1 = R_2$ and $R_1 \neq R_2$ for roles. In particular, $A_1^{\mathcal{I}}(d) = A_2^{\mathcal{I}}(d)$ also covers the case where both values are undefined. This differs from the definitions used in [HN90] and computational linguistics in that we do not require that both attributes have to be defined on d. However, these definitions can be expressed using our constructs:

$$(A_1 = A_2) \sqcap (A_1 : \top) \sqcap (A_2 : \top)$$
$$(A_1 \neq A_2) \sqcap (A_1 : \top) \sqcap (A_2 : \top)$$

As abbreviations for these two expressions we propose $A_1 \overset{\downarrow}{=} A_2$ and $A_1 \overset{\downarrow}{\neq} A_2$, where the downarrow is meant to express the condition "is defined".

4 Role Forming and Attribute Forming Operators

Similar as for concepts our terminological logic provides a variety of role forming and attribute forming operators:

Concrete Form	**Abstract Form**	**Semantics**
(and $R_1 \dots R_n$)	$R_1 \sqcap \dots \sqcap R_n$	$R_1^{\mathcal{I}} \cap \dots \cap R_n^{\mathcal{I}}$
(or $R_1 \dots R_n$)	$R_1 \sqcup \dots \sqcup R_n$	$R_1^{\mathcal{I}} \cup \dots \cup R_n^{\mathcal{I}}$
(not R)	$\neg R$	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$
identity	id	$\{(d,d) \mid d \in \Delta^{\mathcal{I}}\}$
(inverse R)	R^{-1}	$\{(d,d') \mid (d',d) \in R^{\mathcal{I}}\}$
(restrict R C)	$R \mid C$	$\{(d,d') \in R^{\mathcal{I}} \mid d' \in C^{\mathcal{I}}\}$
(compose $R_1 \dots R_n$)	$R_1 \circ \dots \circ R_n$	$R_1^{\mathcal{I}} \circ \dots \circ R_n^{\mathcal{I}}$
(domrange C_1 C_2)	$C_1 \times C_2$	$C_1^{\mathcal{I}} \times C_2^{\mathcal{I}}$
(trans R)	R^+	$\bigcup_{n\geq 1}(R^{\mathcal{I}})^n$
(transref R)	R^*	$\bigcup_{n\geq 0}(R^{\mathcal{I}})^n$
(inverse A)	A^{-1}	$\{(A^{\mathcal{I}}(d), d) \mid d \in dom\, A^{\mathcal{I}}\}$
(restrict A C)	$A \mid C$	$A^{\mathcal{I}} \mid_{C^{\mathcal{I}}}$
(compose $A_1 \dots A_n$)	$A_1 \circ \dots \circ A_n$	$A_1^{\mathcal{I}} \circ \dots \circ A_n^{\mathcal{I}}$

Notice that the inverse of an attribute is a role, but in general not an attribute. The range restriction $R \mid C$ can be seen as an abbreviation for $R \sqcap (\top \times C)$. Similarly, a domain restriction on the role R could be expressed as $R \sqcap (C \times \top)$.

Examples: The role *daughter* can be defined as

$$\texttt{female\_relative} \sqcap \texttt{child};$$

the role *successor* can be defined as

$$(\text{inverse } \texttt{predecessor}).$$

5 Terminological Axioms

The terminological axioms (definitions, specializations, and restrictions) are used to specify the knowledge about the world or a part of the world. A set of terminological axioms specifies a terminology $\mathcal{T}$. It selects from all possible interpretations of the language the models of $\mathcal{T}$, i.e., the interpretations satisfying the axioms of $\mathcal{T}$ as described below.

Concrete Form	**Abstract Form**	**Semantics**
(defconcept CN C)	$CN \doteq C$	$CN^\mathcal{I} = C^\mathcal{I}$
(defrole RN R)	$RN \doteq R$	$RN^\mathcal{I} = R^\mathcal{I}$
(defattribute AN A)	$AN \doteq A$	$AN^\mathcal{I} = A^\mathcal{I}$
(defprimconcept CN C)	$CN \sqsubseteq C$	$CN^\mathcal{I} \subseteq C^\mathcal{I}$
(defprimrole RN R)	$RN \sqsubseteq R$	$RN^\mathcal{I} \subseteq R^\mathcal{I}$
(defprimattribute AN R)	$AN \sqsubseteq R$	$AN^\mathcal{I} \subseteq R^\mathcal{I}$
(defdisjoint $CN_1 \ldots CN_n$)	$CN_1 \parallel \ldots \parallel CN_n$	$CN_i^\mathcal{I} \cap CN_j^\mathcal{I} = \emptyset, i \neq j$

Usually the following restrictions are imposed on terminologies. Any name should appear only once as a left hand side of an axioms, and disjointness axioms should only contain names of primitive concepts.

An alternative way of expressing disjointness could be the use of disjointness groups in the definition of primitive concepts. In this case the introduction of primitive concepts would be of the form $CN \sqsubseteq C/g_1, \ldots, g_n$, where the g_i's are names of disjointness groups. Two different primitive concepts must have disjoint extensions if a disjointness group occurs in the definitions of both concepts.

In the abstract form there is no syntactic distinction between definitions of concepts, roles, and attributes. One possibility to distinguish between concepts, roles, and attributes could be to group the definitions, as done in the following example.

Example (our introductory example in formal notation):
Attributes:
$\texttt{sex} \sqsubseteq \top \times \top$
Roles:
$\texttt{child} \sqsubseteq \top \times \top$
Concepts:

$\texttt{Person} \sqsubseteq \texttt{sex} : \{\texttt{Male, Female}\}$
$\texttt{Woman} \sqsubseteq \texttt{Person} \sqcap \texttt{sex} : \texttt{Female}$
$\texttt{Man} \sqsubseteq \texttt{Person} \sqcap \texttt{sex} : \texttt{Male}$
$\texttt{Parent} \doteq \texttt{Person} \sqcap \exists\texttt{child} : \texttt{Person} \sqcap \forall\texttt{child} : \texttt{Person}$
$\texttt{Mother} \doteq \texttt{Parent} \sqcap \texttt{sex} : \texttt{Female}$
$\texttt{Father} \doteq \texttt{Parent} \sqcap \texttt{sex} : \texttt{Male}$
$\texttt{Mother\_with\_many\_children} \doteq \texttt{Mother} \sqcap \geq 3\texttt{child} : \texttt{Person}$
$\texttt{Father\_with\_sons\_only} \doteq \texttt{Father} \sqcap (\texttt{child} = \texttt{son})$.

Please note that the disjointness axiom Woman $\|$ Man would be redundant since disjointness of woman and man is a consequence of the fact that sex is an attribute and male and female are individuals which are interpreted with unique name assumption.

6 Assertional Axioms

In order to fill our world with objects we allow for assertional axioms which have the following forms.

Concrete Form	**Abstract Form**	**Semantics**
$(C\ ON)$	$ON \in C$	$ON^{\mathcal{I}} \in C^{\mathcal{I}}$
$(R\ ON\ ON')$	$\langle ON, ON' \rangle \in R$	$(ON^{\mathcal{I}}, ON'^{\mathcal{I}}) \in R^{\mathcal{I}}$
$(A\ ON\ ON')$	$\langle ON, ON' \rangle \in A$	$ON^{\mathcal{I}} \in dom\, A^{\mathcal{I}} \wedge A^{\mathcal{I}}(ON^{\mathcal{I}}) = ON'^{\mathcal{I}}$

Examples:
$\texttt{John} \in \texttt{Father}$
$\texttt{Mary} \in \texttt{Woman}$
$\langle\texttt{John}, \texttt{Tom}\rangle \in \texttt{child}$.

7 Services

Now we are able to give a formal specification of the services mentioned in the introduction.

1. Satisfiability of a concept C in a terminology $\mathcal{T}$:
   Does there exist a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$?
   ($\texttt{Man} \sqcap \texttt{Woman}$ is not satisfiable.)

2. Subsumption within a terminology $\mathcal{T}$:
   $C \sqsubseteq_{\mathcal{T}} D$ iff in all models $\mathcal{I}$ of $\mathcal{T}$: $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
   (e.g. $\texttt{Mother} \sqsubseteq_{\mathcal{T}} \texttt{Woman}$).

3. Equivalence of concepts within a terminology $\mathcal{T}$:
   $C \approx_{\mathcal{T}} D$ iff in all models $\mathcal{I}$ of $\mathcal{T}$: $C^{\mathcal{I}} = D^{\mathcal{I}}$

4. Classification of C in $\mathcal{T}$:
   For a given concept C, find all minimal (w.r.t. the subsumption relation) concepts D in $\mathcal{T}$ such that $C \sqsubseteq_{\mathcal{T}} D$.

5. Find the smallest binary relation on the concepts in $\mathcal{T}$ such that its transitive closure is the subsumption relation (modulo $\approx_{\mathcal{T}}$).

6. Consistency of the represented knowledge.
   Does there exist a model $\mathcal{I}$ of the terminological and assertional axioms ?

7. What facts are deducible from the knowledge ?
   A fact α is deducible from the knowledge iff all models of the terminological and assertional axioms satisfy α. In particular, if α is of the form $ON \in C$, then we talk about instantiation.

8. Realization.
   Given an object *ON* occurring in an assertional axiom. Which are most specific concepts of $\mathcal{T}$ w.r.t. the subsumption relation *ON* is instance of?

9. Retrieval.
   Given an concept C. Which objects occurring in the assertional axioms are instances of C ?

With this formalization of our services we can develop procedures or algorithms for the services and prove their correctness, completeness, complexity, decidability; see for example [Sc89, Pa89b, SS88, Ne88, Ne90, Pa89a, HN90, Ho90, Ba91, DL+91a, DH+91, HB91, BH91, DL+91b].

References

[Ba91] F. Baader. *Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles.* DFKI Research Report RR-90-13, DFKI, Postfach 2080, W-6750 Kaiserslautern, Germany. To appear in *Proceedings of IJCAI '91.*

[BH91] F. Baader, P. Hanschke. *A Schema for Integrating Concrete Domains into Concept Languages.* DFKI Research Report RR-91-10, DFKI, Postfach 2080, W-6750 Kaiserslautern, Germany. To appear in *Proceedings of IJCAI '91.*

[BS85] R. J. Brachman, J. G. Schmolze. "An Overview of the KL-ONE knowledge representation system." *Cognitive Science,* 9(2):171-216, April 1985.

[DH+91] F. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, W. Nutt. *The Complexity of Existential Quantification in Concept Languages.* DFKI Research Report RR-91-02, DFKI, Postfach 2080, W-6750 Kaiserslautern Germany.

[DL+91a] F. Donini, M. Lenzerini, D. Nardi, W. Nutt. "The Complexity of Concept Languages." In J. A. Allan, R. Fikes, E. Sandewall (editors), *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, Mas., 1991.

[DL$^+$91b] F. Donini, M. Lenzerini, D. Nardi, W. Nutt. "Tractable Concept Languages." To appear in *Proceedings of IJCAI '91.*

[Ho90] B. Hollunder. "Hybrid Inferences in KL-ONE-based Knowledge Representation Systems." In *Proceedings of the 14th German Workshop on Artificial Intelligence*, pp. 38–47, Eringerfeld, Germany, 1990.

[HB91] B. Hollunder, F. Baader. "Qualifying Number Restrictions in Concept Languages." In J. A. Allan, R. Fikes, E. Sandewall (editors), *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, Mas., 1991.

[HN90] B. Hollunder, W. Nutt. *Subsumption Algorithms for Concept Languages.* DFKI Research Report RR-90-04, DFKI, Postfach 2080, W-6750 Kaiserslautern, Germany.

[Ne90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, Lecture Notes in Artificial Intelligence, LNAI 422, Springer Verlag, 1990.

[Ne88] B. Nebel. "Computational complexity of terminological reasoning in BACK." *Artificial Intelligence*, 34(3):371–383, 1988.

[Pa89a] P. Patel-Schneider. "A four-valued Semantics for Terminological Logics." *Artificial Intelligence*, 38(3):319-351, 1989.

[Pa89b] P. Patel-Schneider. "Undecidability of Subsumption in NIKL." *Artificial Intelligence*, 39(2):263-272, 1989.

[Sc89] M. Schmidt-Schauß. "Subsumption in KL-ONE is undecidable." In R. J. Brachmann, H. J. Levesque, R. Reiter (editors), *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pp. 421–431, Toronto, Ont., 1989.

[SS88] M. Schmidt-Schauß, G. Smolka. "Attributive Concept Descriptions with Unions and Complements". In *Artificial Intelligence*, 47, 1991.

List of participants

- **Jürgen Allgayer**
  Universität des Saarlandes
  Im Stadtwald 15
  D-6600 Saarbrücken
  Germany
  allgayer@cs.uni-sb.de

- **Guiseppe Attardi**
  Dipartimento di Informatica
  Corso Italia 40
  I-56125 Pisa
  Italy
  attardi@di.unipi.it

- **Franz Baader**
  DFKI
  Postfach 2080
  Erwin-Schrödinger-Str.
  D-6750 Kaiserslautern
  Germany
  baader@dfki.uni-kl.de

- **Howard W. Beck**
  Computer and Information Sciences
  460 CSE
  University of Florida
  Gainsville, FL 32611
  USA
  hwb@beach.cis.ufl.edu

- **Sonia Bergamaschi**
  CIOC-CNR
  Universita di Bologna
  viale Risorgimento 2
  I-40136 Bologna
  Italy
  sonia@deis64.cineca.it

- **Ronald J. Brachman**
  AT & T Bell Labs
  600 Mountain Ave. 3C-439
  Murray Hill, NJ 07974
  USA
  rjb@research.att.com

- **Amedeo Cappelli**
  Istituto di Linguistica Computazionale
  Via della Faggiola 32
  I-56100 Pisa
  Italy
  sistemi@icnucevm.cnuce.cnr.it

- **Francesco M. Donini**
  Dipartimento di Informatica e Sistemistica
  Universita di Roma "La Sapienza"
  Via Salaria 113
  I-00198 Roma
  Italy
  donini@vaxrma.infn.it

- **Enrico Franconi**
  IRST - Istituto per la Ricera
  Scientifica e Tecnologica
  I-38050 Povo TN
  Italy
  franconi@irst.it

- **Manfred Gehrke**
  SIEMENS AG
  ZFE F2 INF 23
  Otto-Hahn-Ring 6
  D-8000 München 83
  Germany
  gehrke@ztivax.siemens.com

- **Nicola Guarino**
  LADSEB-CNR
  Corso Stati Uniti 4
  I-35020 Padova
  Italy
  Guarino@ladseb.pd.cnr.it

- **Jochen Heinsohn**
  DFKI
  Stuhlsatzenhausweg 3
  D-6600 Saarbrücken 11
  Germany
  heinsohn@dfki.uni-sb.de

- **Carsten Kindermann**
  Technische Universität Berlin
  FR 5-12
  Franklinstraße 28/29
  D-1000 Berlin 10
  Germany
  Carsten.Kindermann@cs.tu-berlin.de

- **Karin Klabunde**
  Philips GmbH
  Forschungslaboratorium Aachen
  Postfach 1980
  D-5100 Aachen
  Germany
  klabunde@philfa.uucp

- **Rüdiger Klein**
  Otto-Brahm-Str 26
  O-1120 Berlin
  Germany
  klein@city.zki-berlin.adw.dbp.de

- **Alfred Kobsa**
  SFB 314
  FB-10 Informatik
  Universität des Saarlandes
  D-6600 Saarbrücken 11
  Germany
  ak@cs.uni-sb.de

- **Hector J. Levesque**
  Department of Computer Science
  University of Toronto
  10 King's College Road
  Toronto, Ont. M5S 1A7
  Canada
  hector@ai.toronto.edu

- **Kai von Luck**
  IBM Deutschland GmbH
  IWBS 7000-75
  Postfach 80 08 80
  D-7000 Stuttgart 80
  Germany
  luck@ds0lilog.bitnet

- **Robert MacGregor**
  USC/ISI
  4676 Admiralty Way
  Marina del Rey, CA 90292
  USA
  macgregor@isi.edu

- **Eric K. Mays**
  IBM Watson Research Center
  P.O. Box 218
  Yorktown Heights, NY 10598
  USA
  emays@ibm.com

- **Deborah L. McGuinness**
  AT&T Bell Laboratories
  600 Mountain Avenue
  Murray Hill, NJ 07974
  USA
  dlm@research.att.com

- **Bernhard Nebel**
  DFKI
  Stuhlsatzenhausweg 3
  D-6600 Saarbrücken 11
  Germany
  nebel@dfki.uni-sb.de

- **Werner Nutt**
  DFKI
  Postfach 2080
  Erwin-Schrödinger-Str.
  D-6750 Kaiserslautern
  Germany
  nutt@dfki.uni-kl.de

- **Lin Padgham**
  Linköping University
  Computer and Information Science Dept.
  S-58183 Linköping
  Sweden
  lin@ida.liu.se

- **Peter F. Patel-Schneider**
  AT&T Bell Labs
  600 Mountain Ave.
  Murray Hill, NJ 07974
  USA
  pfps@research.att.com

- **Christof Peltason**
  Technische Universität Berlin
  FR 5-12
  Franklinstraße 28/29
  D-1000 Berlin 10
  Germany
  peltason@cs.tu-berlin.de

- **Bernhard Pfahringer**
  Austrian Research Institute
  for Artificial Intelligence
  Schottengasse 3
  A-1010 Vienna
  Austria
  bernhard@ai-vie.uucp

- **Udo Pletat**
  IBM Deutschland GmbH
  IWBS 7000-75
  Postfach 80 08 80
  D-7000 Stuttgart
  Germany
  pletat@ds0lilog.bitnet

- **H.-J. Profitlich**
  Deutsches Forschungszentrum
  für Künstliche Intelligenz GmbH
  Stuhlsatzenhausweg 3
  D-6600 Saarbrücken 11
  Germany
  profi@dfki.uni-sb.de

- **Joachim Quantz**
  Technische Universität Berlin
  FR 5-12
  Franklinstraße 28/29
  D-1000 Berlin 10
  Germany
  jjq@cs.tu-berlin.de

- **Klaus Schild**
  Technische Universität Berlin
  FR 5-12
  Franklinstraße 28/29
  D-1000 Berlin 10
  Germany
  ks@cs.tu-berlin.de

- **Albrecht Schmiedel**
  Deutsches Herzzentrum Berlin
  Projektgruppe Medizin Informatik
  Voltastraße 5
  D-1000 Berlin 65
  Germany
  atms@cs.tu-berlin.de

- **Roland Seiffert**
  IBM Deutschland GmbH
  IWBS 7000-75
  Postfach 80 08 80
  D-7000 Stuttgart 80
  Germany
  seiffert@ds0lilog.bitnet

- **Gert Smolka**
  DFKI
  Stuhlsatzenhausweg 3
  D-6600 Saarbrücken 11
  Germany
  smolka@dfki.uni-sb.de

- **Luca Spampinato**
  Quinary S.p.A.
  Via Crivelli 15/1
  I-20123 Milano
  Italy
  ls@quinary.uucp

- **William R. Swartout**
  USC/ISI
  4676 Admiralty Way
  Marina del Rey, CA 90292
  USA
  swartout@isi.edu

- **Wolfgang Wahlster**
  Universität d. Saarlandes
  Fachbereich 10
  Bau 36
  D-6600 Saarbrücken
  Germany
  wahlster@cs.uni-sb.de

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen oder die aktuelle Liste von erhältlichen Publikationen können bezogen werden von der oben angegebenen Adresse.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of currently available publications can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-90-01
Franz Baader: Terminological Cycles in KL-ONE-based Knowledge Representation Languages
33 pages

RR-90-02
Hans-Jürgen Bürckert: A Resolution Principle for Clauses with Constraints
25 pages

RR-90-03
Andreas Dengel, Nelson M. Mattos: Integration of Document Representation, Processing and Management
18 pages

RR-90-04
Bernhard Hollunder, Werner Nutt: Subsumption Algorithms for Concept Languages
34 pages

RR-90-05
Franz Baader: A Formal Definition for the Expressive Power of Knowledge Representation Languages
22 pages

RR-90-06
Bernhard Hollunder: Hybrid Inferences in KL-ONE-based Knowledge Representation Systems
21 pages

RR-90-07
Elisabeth André, Thomas Rist: Wissensbasierte Informationspräsentation:
Zwei Beiträge zum Fachgespräch Graphik und KI:
1. Ein planbasierter Ansatz zur Synthese illustrierter Dokumente
2. Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen

24 pages

RR-90-08
Andreas Dengel: A Step Towards Understanding Paper Documents
25 pages

RR-90-09
Susanne Biundo: Plan Generation Using a Method of Deductive Program Synthesis
17 pages

RR-90-10
Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, Jörg H. Siekmann: Concept Logics
26 pages

RR-90-11
Elisabeth André, Thomas Rist: Towards a Plan-Based Synthesis of Illustrated Documents
14 pages

RR-90-12
Harold Boley: Declarative Operations on Nets
43 pages

RR-90-13
Franz Baader: Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles
40 pages

RR-90-14
Franz Schmalhofer, Otto Kühn, Gabriele Schmidt: Integrated Knowledge Acquisition from Text, Previously Solved Cases, and Expert Memories
20 pages

RR-90-15
Harald Trost: The Application of Two-level Morphology to Non-concatenative German Morphology
13 pages

International Workshop on Terminological Logics

D-91-13
Document

Organizers: Bernhard Nebel, Christof Peltason, Kai von Luck