DFKI-Workshop

Knowledge Representation Techniques

Kaiserslautern, July 8, 1993

Proceedings

Knut Hinkelmann, Armin Laux (Eds.)

Contents

DFKI-Workshop

Knowledge Representation Techniques

Kaiserslautern, July 8, 1993

Program

9:00 - 10:30 Session 1, *Chair: B. Nebel*

Towards Evolvable Knowledge Representation for Industrial Applications
H. Boley

RanDom: Representation Language for *Dom*ain Theories
B. Bachmann

Representation of Plan Libraries
J. Köhler

1:00 - 12:30 Session 2, *Chair: H.-J. Bürckert*

WINO/TACOS: Terminological Knowledge Representation
B. Hollunder, F. Baader

An Empirical Analysis of Terminological Representation Systems
H.-J. Profitlich, J. Heinsohn, D. Kudenko, B. Nebel

Knowledge Representation and Reasoning in TOOCON
H. Wache

13:30 - 15:30 Session 3, *Chair: H. Boley*

KR for Natural Language Processing Agents
E. Hinkelman

Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra
B. Nebel, H.-J. Bürckert

ALCP: Handling Uncertainty in Description Logics
J. Heinsohn

Knowledge Representation in the Rule-based Multi-Agent System MAGSY
K. Fischer

16:00 - 17:00 Final Discussion

Participants

Andreas Abecker, Bernd Bachmann, Harold Boley, Stefan Busemann, Martin Buchheit, Hans-Jürgen Bürckert, Alastair Burt, Jörg Denzinger, Fredj Dridi, Klaus Fischer, Matthias Fuchs, Philipp Hanschke, Jochen Heinsohn, Elizabeth Hinkelman, Knut Hinkelmann, Rainer Hoch, Bernhard Hollunder, Jana Köhler, Armin Laux, Michael Malburg, Manfred Meyer, Bernhard Nebel, Hans-Jürgen Profitlich, Anne Schöller, Werner Stein, Donald Steiner, Frank Steinle, Holger Wache, Jörg Würtz

Preface

The internal DFKI Workshop on "Knowledge Representation Techniques" was held in Kaiserslautern on July 8, 1993, and was attended by a total of about 30 participants from both sites.

The intent of the workshop, as recommended by the Scientific Advisory Board of the DFKI, was to "point out the differences between representation techniques as used in DFKI" and "how they relate to the work of the groups".

The workshop's principal goal was to improve communication and collaboration between developers and users of knowledge representation (KR) systems within DFKI in order to support extensive exchange of experience with knowledge representation systems currently under development and to discuss requirements coming from practical applications. A long-term aim of the workshop was to encourage the use of knowledge representation systems developed at DFKI. To this end, a balanced program of ten research presentations by both developers and users was organized. A main topic of the final discussion was how to bridge the gap between research and utilization of knowledge representation techniques.

There were 10 talks in three sessions. Terminological knowledge representation was the topic of a majority of the talks. They dealt with their efficiency, their extension for special-purpose inferences, and their integration with other reasoning formalisms, but also with additional requirements for particular applications. Other talks address knowledge representation from the point of view of particular applications: natural language processing, multi-agent systems, integration of knowledge sources, temporal reasoning, case-based reasoning, and knowledge base maintenance.

Harold Boley from the VEGA project opened the workshop arguing that, in order to raise the industrial value of knowledge assets, evolvability of knowledge bases should become an additional design criterion for declarative knowledge representation languages. Bernd Bachmann (IMCOD) presented the language RANDOM for the conceptualization of ontologies, which supports unification of heterogeneous terminologies on a semantic level. Jana Köhler (PHI) demonstrated, how terminological logics can be incorporated into a hybrid representation formalism for case bases, making the logical formalization of retrieval and update operations possible.

Bernhard Hollunder gave a short introduction into terminological knowledge representation and presented language extensions examined in the TACOS project. Holger Wache (TOOCON) reported about experiences and requirements from using hybrid knowledge representation and presented an architecture for a prototypical configuration system[1]. Hans-Jürgen Profitlich (WIP) discussed results from an empirical analysis of the classification algorithm of six terminological representation systems.

[1] The paper of this talk has not been included in the proceedings.

In the last session, Elizabeth Hinkelman (DISCO) described the most important data structures and reasoning techniques to support natural language processing in DISCO's appointment scheduling application. Bernhard Nebel introduced the ORD-Horn subclass, which is the unique maximal tractable subclass of Allen's interval algebra for reasoning about temporal relations (among the subclasses that contain all basic relations). Jochen Heinsohn (WIP) presented the language $\mathcal{ALCP}$, a probabilistic extension of terminological logics for dealing with uncertain knowledge. In the last talk, Klaus Fischer described the use of the forward chaining rule interpreter for representing the knowledge of an individual agent and the communication between agents in the multi-agent scenario of the AKA-MOD project.

A principal topic of the final discussion was exactly what requirements a KR system should attempt to satisfy. Selection of requirements is complicated by the wide range of possible applications. In the past, this has led to development of various KR systems at DFKI, which are excellent for specific functions. However, any substantive application also has several requirements to be met simultaneously. Some of these requirements are efficiency requirements rather than functionality requirements.

Both users and developers of KR systems agreed that there cannot be a single KR system which is adequate for *all* AI systems. Terminological knowledge representation systems are a focus of research in many projects at DFKI. But they are not satisfactory for non-toy applications. At DFKI there is a great effort in extending the expressiveness of terminological logics and integrating them with other representation and reasoning paradigms. These extensions are driven by various applications of terminological systems: In the TACOS project non-monotonic approaches and modal logics are examined in cooperation with the application scenario of AKA-MOD. The representation of autonomous agents also required the use of more procedural knowledge, leading to the rule-based system MAGSY. In constrast, the rule component of the hybrid representation and compilation laboratory CoLab, which has been developed in the ARC-TEC project, is a more declarative logical system. CoLab's concrete domains and its constraint solver over hierarchical domains have also been necessary for a configuration application. A further extension of terminological logics is the probabilistic reasoning developed in WIP. But there are other projects that do not use terminological systems. RanDom, for instance, is a new language for representing ontologies that is more expressive than terminological systems. Besides the definition of concepts it is also possible to define relations.

Although knowledge-based systems are often not applied stand-alone, several methods were identified for increasing the range of possible applications that should be addressed by every KR system:

- providing a judiciously selected set of basic facilities
- allowing for extensions by the user
- allowing for embedding of the KR system into application systems

In particular, integration of the KR system into a host programming language might facilitate all three of these tasks. It would enable communication between the KR system and application software, support combination and communication of several different KR tools within an AI application, and provide procedural means for addressing any residual KR or application problems.

Even with the development of upgradable KR tools using this design, there remain at least three problems. First, KR users are not always able to anticipate needed KR facilities,

since they often begin with little practical experience in the application area. There may not be time to develop a complete model of knowledge to be represented, develop the KR facilities, and then develop the application. Second, selection and comparison of KR systems is often hindered by the lack of standard descriptions for KR system services. The workshop participants agreed that these problems could be reduced through frequent and timely communication between developers and users, with the result that powerful intelligent KR systems will be developed. Existing KR systems should be used and adapted wherever this is possible. Third, declarative knowledge representation and integration into a conventional software environment with procedural aspects may be conflicting. This may be overcome by shifting procedural aspects into separate components.

One suggestion for improving communication between users and developers was to establish a permanent "representation forum" in the form of a mailing list or a newsgroup. It would then be possible for KR system users to describe their problems in KR, and for the various KR system development groups to try to solve them. This can be seen as a kind of competition in which users choose the best-fitting solution.

Finally, all participants agreed on the fact that developing KR systems is an essential part of basic research in AI. Thus, there should be different tools within DFKI which are appropriate for different AI applications, and for different tradeoffs between state-of-the-art functionality and efficient implementations. However, if these systems are comparable with each other, and if there is a representation forum as described above, the reuse and integration of existing KR systems into AI applications should be essentially simplified.

Knut Hinkelmann, Armin Laux

1

Towards Evolvable Knowledge Representation for Industrial Applications

Harold Boley[1]

ABSTRACT For permitting the construction of large, industrial knowledge bases, represention languages should have a simple nucleus (DATALOG) facilitating knowledge evolution, i.e. exploration and validation. Extensions via a sorted Horn language with finite domains (a homogeneous version of the hybrid language in our compilation lab) should preserve evolvability of our sharable materials knowledge base.

1.1 Introduction

The AI field of knowledge representation (KR) today not only deals with KR languages and techniques but also with the actual construction of large knowledge bases (KBs). A computerized KB formalizing a non-trivial part of the huge body of human knowledge will be mandatory for both approaches to AI, the 'cognitive' and the 'engineering' (our own) one: we cannot really use an AI system as a model of some behavior or as an assistant for some expert if the knowledge represented comprises little more than the prototype entries needed for hasty demonstrations and page-limited publications.

Unfortunately, there is an apparent reciprocal relationship between the size of a KB and the richness of its KR language, since complex language features can be better controlled in small KBs, while realistic-sized KBs can be better managed in simple languages. This stems not only from the computational trade-off between efficiency and expressiveness but also from the organizational problem of maintaining a large KB represented in a variety of complex formats rather than in a uniform and simple one.

A large KB cannot be constructed at once, but will call for a process of knowledge **evolution** that validates and explores a given KB state before extending it in an interactive feed-back loop. This requires KB analysis tools which will again profit from a uniform, simple KR language to avoid being trapped in the case analysis of interdependent language features rather than advancing in the interpretation of knowledge content. Another consideration in striving for realistic KR languages is the fact that industrially successful software tools (AI and non-AI) often are simplified forms of their academic prototypes.

In this setting we propose an R&D approach starting with a simple KR language and its evolution tools, constructing an industrially relevant KB, evolving and extending the KB, while extending the language only as determined by the application and evolution concerns.

[1] Project VEGA, DFKI Kaiserslautern, email: boley@dfki.uni-kl.de

1.2 The DATALOG Nucleus

The explicit representation of knowledge typical for AI research may well become widespread also for industrial applications. Relational database systems already have demonstrated the practical use of such a declarative, machine-independent representation. The representation language of (pure) DATALOG, i.e. pure PROLOG without constructors, underlying – deductive – database systems, thus provides a simple and natural starting point for step-by-step extensions towards industrial knowledge representation.

There is another reason, perhaps even more important, for starting language development with a DATALOG nucleus: database maintenance and mining systems analyzing large DATALOG volumes have already proved their practical usefulness by discovering cost-relevant information. This is indicated, for instance, by Piatetsky-Shapiro's publications on successful applications of KDD (knowledge discovery in databases) systems and by commercial database-mining tools such as Lockheed Corporation's Recon, which, e.g., analyzed 40 MB of data on printed circuit board manufacturing, extracting knowledge used to evaluate and adjust the manufacturing process. In the DATALOG nucleus, algorithms for such a global analysis and evolution of KBs can decide particularly many interesting properties.

1.3 Evolvable DATALOG Extensions

In the VEGA project, the first planned layer of DATALOG extensions provides (1) constructor symbols, (2) finite-domain constraints, and (3) intensional sorts. Variants of all of these have an obvious practical relevance, namely (1) as complex domains in non-standard databases, (2) as alternatives to combinatorial algorithms, and (3) as models for data dictionaries. The resulting sorted Horn language with finite domains will be a homogeneous version of our hybrid CoLab (compilation laboratory) language: (a) Knowledge in the 'ABox' and 'TBox' parts is represented in a uniform fashion. (b) Domains and sorts are usable both as variable annotations and as unary predicates.

For extending evolvability from databases to KBs we are studying the scalability of existing static-analysis and machine-learning algorithms to the above language extensions. Further linguistic and algorithmic extensions will certainly arise via cross-fertilization with the development and maintenance of our realistic declarative demonstrator KB (see below).

Candidates of useful language constructs include higher-order notation (predicate variables and explicit 'subsumes' predicate), strong negation (in combination with negation as failure), modules (as premises of implicational goals), and conditional answers (for abduction).

1.4 An Application KB

The sharable VEGA demonstrator KB is currently being developed using a portable subset of CoLab's KR language as a preliminary representation language. It accumulates know-how in the engineering domain of materials (as occurring in electronics/computers) for supporting the complementary tasks of construction and recycling.

Such a KB can reflect the continuous improvement and multiple use of the materials represented (mainly plastics and composite materials), thus providing a good testbed for knowledge evolution and sharing. Our own planned application of the materials KB, in collaboration with the IVW (a research institute for industrial-quality composite materials), is an

expert assistant selecting materials for composite constructions (with properties including those relevant to the challenge of recyclability).

Concerning evolution of the materials KB, we are considering the exploration of, e.g., material-property links (thus supporting the expert assistant) and possible material-material substitutions as well as the validation of, e.g., new composite materials against environmental requirements. VEGA envisages an interactive knowledge-evolution loop in which the knowledge engineer can feed back validated findings (e.g. a recyclable composite material) into the KB. A design criterion characteristic to representation languages of such 'closed-loop' (Wrobel) learning systems is 'evolutionary closure', the power to express all new findings in the original KR language.

1.5 Implementation, Availability, and Conversion

For experimenting with the operational semantics of the language extensions, they will be specified by a definitional interpreter. Then, for obtaining language efficiency, we use a compilative implementation approach extending WAM technology both 'upward' (source-to-source translation) and 'downward' (low-level emulators). Regarding evolution efficiency, we plan to implement a library of algorithms of decreasing complexity for decreasing language subsets; so, restricting one's KR-language use (to DATALOG, in the extreme), it will be possible to evolve larger KBs. Our implementation task is supported by the fact that certain inference algorithms of representation languages such as LDL (bottom-up algorithms) and KL-ONE (classification algorithms) can be reused for KB evolution.

The language compilation and evolution tools as well as the demonstrator KB will be available as public-domain software as new versions are being released (presently CoLab and a 1K-lines pure-PROLOG KB on the chemical elements).[2]

In order to evolve real-world data it is planned to write a converter from the industrial SQL standard for relational databases to the VEGA KR language. We are also studying intertranslation options between the VEGA language and knowledge-interchange standards such as KIF. Because of its DATALOG nucleus the language should be easy to learn by users in industry accustomed to relational databases; this is supported by user manuals, sample KBs, and tutorial dialog scripts.

1.6 Conclusion

In a nutshell, we aim to supplement the ordinary KB **querying** via inference engines (for expert problem solving) with KB **maintenance** via an evolution engine (for quality improvement). This leads to additional language-design criteria concerning KB evolvability, which are related to, and combinable with, Swartout's additional design criteria concerning KB explainability. It is clear that KB evolvability, like explainability, also raises the industrial value of knowledge assets: only if a company's know-how representation can be re-molded simultaneously with changes of internal and external reality, will the effort of putting it into the machine be worth the price.

[2] Ftp to `serv-200.dfki.uni-kl.de` as user `anonymous` and cd into `PROJECTS/vega`

2

RANDOM — *R*epresentation L*an*guage for *Dom*ain Theories

Bernd Bachmann[1]

ABSTRACT The IMCOD project requires the coupling of heterogenous systems that have not been designed for their later integration. We present an overview of the knowledge representation language RANDOM that provides capabilities for representing domain ontologies. Individual ontologies may then be used to gain semantic unification between diverging conceptualizations of individual systems.

2.1 Motivation

The theoretical aspects and practical issues of exchanging knowledge and data in heterogeneous environments are motivated by various disciplines and research issues, e.g. environments for concurrent engineering, integration of XPSs and EDP systems, enterprise integration, etc. Despite its obvious commercial import, technologies for enabling automatic communication and exchange of knowledge and data on a semantic level are still in its infancy.

One of the problems that arises when coupling heterogeneous systems can be addressed as *semantic unification*. Each system is supposed to be built on the basis of an independent conceptualization and model that are tailored towards its reasoning and representation capabilities and requirements. Though the conceptualization of two such systems may be represented within a uniform representation language, say first order predicate logic (FOPL), the problem remains how different expressions in both conceptualization which denote the same objects and relationships in the real world can be unified and, therefore, transformed into each other.

The following approach for semantic unification is determined by the requirements of the recently started project IMCOD [BBK+92] (*I*ntelligent *M*anager for *Co*mprehensive *D*esign) which is concerned with developing a centrally managing design system that supports a design manager while performing a product design by integrating various local knowledge and information sources called local experts. Local experts may be realized by expert systems or conventional EDP tools like CAD systems, databases, etc. and contribute a single functionality to the overall system, i.e. they act as experts for geometry, material, manufacturing, recycling, etc., cf figure 2.1.

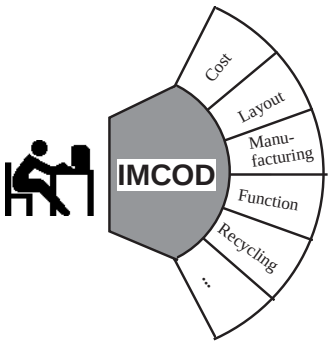

FIGURE 2.1. The IMCOD System

[1]Project IMCOD, DFKI Kaiserslautern, email: bachmann@dfki.uni-kl.de

2.2 Ontologies and Communication Layers

Knowledge that is exchanged between knowledge based systems is interpreted according to a model that is individual for each participant. We introduce the term *ontology* to capture this fact. Philosophically, an ontology describes the way the universe is carved up[2]. In the context of knowledge–based systems an ontology can be regarded as the definition of the objects, relations, and functions that form the basis for the conceptualization and model of an expert system. However, an ontology does not only include the base set of a conceptualization in the logical sense (cf [GN87]) but also makes explicit (some of) the knowledge that forms the intrinsic properties of the elements of the base sets. Consequently, if ontologies are represented in first order predicate logic they can be regarded as theories of a specific domain on which the system relies.

For example, the right side of figure 2.2 shows the definition of the relation *leftneighbour* in an ontology about the geometry of turning workpieces [Dup93]. It is defined with necessary and sufficient conditions between two *RSGO*s[3] and holds if the two RSGOs are located to the left of each other and have a common edge. Hence, the term *leftneighbour* as the content of a message must only be interpreted according to this restricted meaning. It rules out that the relation can be used as a relationship between building sites or that it is used in the weaker sense of *left–of*.

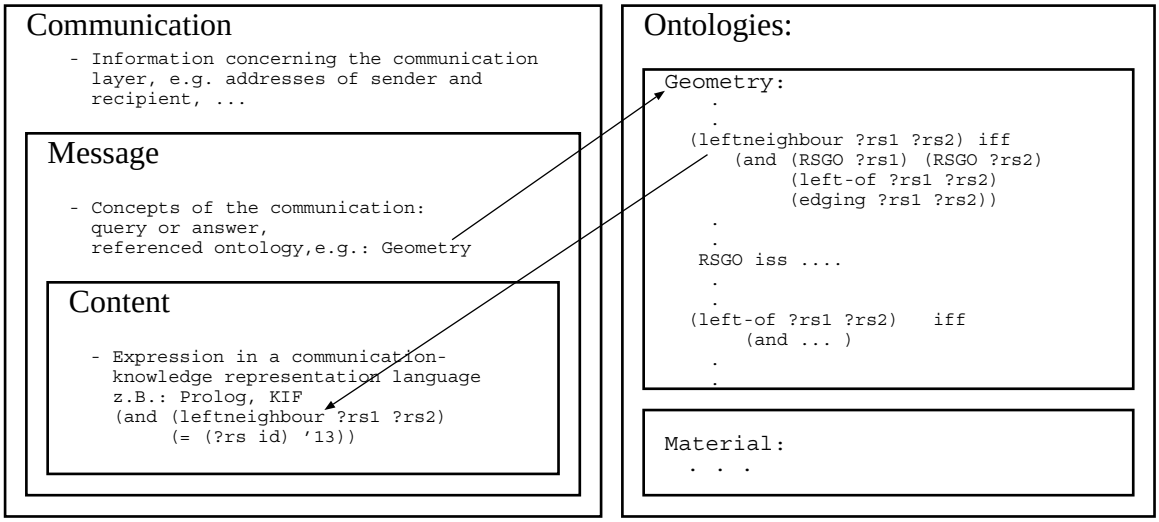

FIGURE 2.2. The Communication Layers and the Underlying Ontologies

In order to show where ontologies play a role in the communication between two expert systems we adapt the model of [PF+92], cf figure 2.2. It consists of three layers: communication, message, and content layer. The communication layer doesn't play an important role in our context. The message layer encompasses information about the message as a whole. For instance, an XPS may have to interpret a message as an assertion of new facts or a query whether the facts are consistent with its current knowledge base. Finally, in the content layer the pure message is represented. Any representation language may be used for that purpose. The crucial issue here is that the symbols (names of objects, relations,

[2] a particular theory about the nature of being or the kinds of existence; cf Webster's dictionary
[3] rotationally symmetrical geometric objects

and functions) in this layer must only be used in accordance with the definitions of the corresponding ontology as mentioned above.

2.3 RanDom

The basic idea to achieve semantic unification in the above scenario requires a procedure in several steps:

1. Represent the ontology of the participating expert systems, i.e. the definition of its base sets as they occur at the interface of such an expert system.
2. Identify epressions that are used in a consistent way in two different ontologies of expert systems that communicate with each other or define explicit mappings between two corresponding expressions, e.g. mappings between object identifiers.
3. From the mappings of these expressions derive the mappings of more complex expressions automatically.

The example in section 2.4 reveals the basic ideas more clearly. RanDom supports this procedure by providing the following functionalities:

- It is a knowledge representation language for defining ontologies (domain theories in FOPL). It provides a homogeneous environment for the representation of objects and relations.
- It allows the hierarchical definitions of ontologies, i.e. more complex ontologies may be built on simpler ones and include them as domain theories.
- Entities in one ontology may be used in definitions of others as shown in figure 2.3.

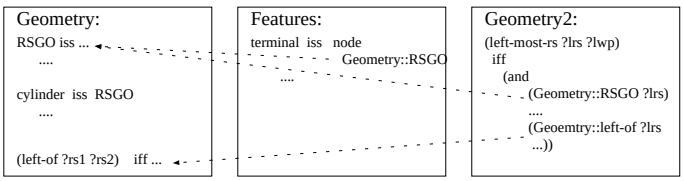

FIGURE 2.3. An example

Consequently, there are some mechanisms that support the definition of corresponding expressions in different ontologies by including a domain theory completely or by the inclusion of single definitions of objects and relations. These form the sets of terms and sentences that are already semantically unified just by the definition of the ontology.

The semantics of a single ontology can be defined by translating the definitions to FOPL. However, the problem remains open how the semantics of diverging conceptualizations can be defined that have to be integrated in a common theory. For that purpose we introduce the notion of a *context* according to [Guh91]. A context provides a logical mechanism to capture the idea of diverging conceptualizations:

- Different contexts could use different languages, i.e. each context has is independent base set of relations, functions, and objects.
- First order theory should be embedded in the sense that within a context the same deductions as in FOPL are possible.

- A symbol might denote different things in different contexts, i.e. the interpretation of an identifier is made with respect to the context in which it is used.

In order to connect contexts with FOPL we introduce a new, two–place operator *ist* (is true in) that takes as arguments a context $\mathcal{C}$ and a formula of FOPL Ψ:

$$ist(\mathcal{C}, \Psi)$$

which itself is a well–formed formula, i.e. the operator is self–embedding. Intuitively, the formula Ψ is true in the specific context $\mathcal{C}$ if Ψ is a well-formed formula in the language that is assigned to $\mathcal{C}$ and Ψ is true as a formula in FOPL under an interpretation and a substitution for the variables. Note however, that one must be careful about the scope of variables, e.g. $\forall x.ist(\mathcal{C}, \Psi)$ is different from $ist(\mathcal{C}, \forall x.\Psi)$ if x is a variable occurring free in Ψ.

The overall objective is to find relationships between two contexts, so called *articulation axioms* [HJK+92] of the form

$$ist(\mathcal{C}_1, \Psi) \Longleftrightarrow ist(\mathcal{C}_2, \Phi)$$

or

$$ist(\mathcal{C}_1, \Psi) \Longrightarrow ist(\mathcal{C}_2, \Phi)$$

which informally denotes that the meaning of Ψ in $\mathcal{C}_1$ is the same or entails the meaning of Φ in $\mathcal{C}_2$.

2.4 An Example

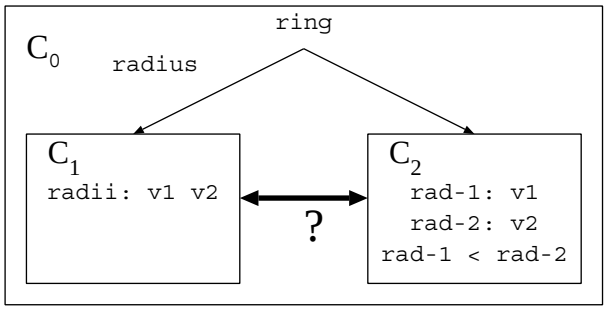

FIGURE 2.4. An example

The example in figure 2.4 illustrates how a ring could be represented in different ways by two expert systems in the domain of turning workpieces. The overall ontology that defines the context $\mathcal{C}_0$ entails the objects *ring* and *radius*. However, there are two diverging specializations. One, $\mathcal{C}_1$, were the radii are represented via a (set valued) relation *radii* and one, $\mathcal{C}_2$, were two functions *rad–1* and *rad–2* are used such that the value of *rad–2* is always greater than the value of *rad–1*. With these two specific conceptualizations the problem arises how the representation of a ring in context $\mathcal{C}_1$ can be automatically translated into a representation in context $\mathcal{C}_2$.

Since $\mathcal{C}_0$ entails $\mathcal{C}_1$ and $\mathcal{C}_2$ we can make variable substitutions (*ring* and *radius*) for variables with respect to $\mathcal{C}_0$ and use these variables in formuals in the subordinated contexts $\mathcal{C}_1$ and $\mathcal{C}_2$. According to our notation above we represent the situation in FOPL enhanced by the *ist*–operator.

$$\begin{aligned} \text{ist}(\mathcal{C}_0, \forall x.\text{ring}(x) \Longrightarrow \quad & \exists y \exists z.\text{radius}(y) \wedge \text{radius}(z) \wedge \\ (\text{ist}(\mathcal{C}_1, \text{radii}(x,y) \wedge \text{radii}(x,z)) \Longleftrightarrow \text{ist}(\mathcal{C}_2, \quad & ((\text{rad–1}(x) = y) \wedge \text{rad–2}(x) = z) \vee \\ & (\text{rad–1}(x) = z) \wedge \text{rad–2}(x) = y)) \\ & \wedge < (\text{rad–1}(x), \text{rad–2}(x)))) \end{aligned}$$

This will result in the following formula which enables the automatic translation of the different conceptualizations at run–time for a specific ring with the radii 5 and 10.

$$\text{ist}(\mathcal{C}_0, \text{ring}(r) \Longrightarrow \text{radius}(5) \wedge \text{radius}(10) \wedge$$
$$(\text{ist}(\mathcal{C}_1, \text{radii}(r, 10) \wedge \text{radii}(r, 5)) \Longleftrightarrow \text{ist}(\mathcal{C}_2, \text{rad–1}(r) = 5 \wedge \text{rad–2}(r) = 10)))$$

The second part of the formula shows an articulation axiom of the from we are seeking for.

2.5 Final Remarks

There are (at least) two questions that have to be further investigated. First, the semantics of the *ist*–operator should be formally defined. There is a formalization in [Guh91] but it is based on an enhanced denotational semantics. Since Kripke structures provide a means to distinguish between two terms that refer to the same object or between two sentences with the same truth values—two situations which seem to be intrinsic in the mentioned situation—a possible–worlds semantics may be more appropriate for contexts. Secondly, it is not clear how much knowledge must be provided and represented for the mapping of primitive entities, e.g. mapping between relations, in order to gain mappings of more complex structures, e.g. first–order formulas.

Finally, it should be emphasized that the discussed approach for semantic unification requires that the conceptualization of the knowledge sources at hand can be represented in FOPL. Also the problem of translating the syntax of RanDom into the idiosyncratic syntax of each knowledge source at run–time has been neglected.

2.6 References

[BBK+92] B. Bachmann, A. Bernardi, C. Klauck, R. Legleitner, and M.M. Richter. IMCOD — Intelligent Manager for Comprehensive Design. In *Workshop Notes (ECAI '92): Concurrent Engineering: Requirements for Knowledge–Based Design Support*, 1992.

[Dup93] J. Dupont. Eine Ontologie für die Arbeitsplanung von Drehteilen. Master's thesis, University of Kaiserslautern, August 1993. in German.

[GN87] M.R. Genesereth and N.J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, 1987.

[Guh91] R.V. Guha. Contexts: A formalization and some applications. MCC Technical Report ACT–CYC–423–91, Microelectronics and Computer Technology Corporation, Advanced Computing Technology, Artificial Intelligence Lab, 3500 West Balcones Center Drive, Austin, TX 78759, November 1991.

[HJK+92] M.H. Huhns, N. Jacobs, T. Ksiezyk, W.-M. Shen, M.P. Singh, and P.E. Cannata. Enterprise information modeling and model integration in Carnot. In Petrie [Pet92], pages 290–299.

[Pet92] C.J. Petrie, editor. *Proc. 1st Int. Conf. on Enterprise Integration Modeling.* MIT Press, 1992.

[PF+92] R.S. Patil, R.E. Fikes, et al. The DARPA knowledge sharing effort: Progress report. In B. Nebel, C. Rich, and W. Swartout, editors, *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 777–788, 1992.

3

Representation of Plan Libraries

Jana Köhler[1]

ABSTRACT

The reuse and modification of previously synthesized artifacts such as plans, designs, proofs and programs to meet new specifications has emerged as an active topic of research in may areas of AI, including *planning*, *scheduling*, *design*, *automated theorem proving* and *software engineering*. One of the key problems which has to be addressed is the representation, organization and maintenance of libraries in which these reusable artifacts can be stored. Current approaches mainly revolve around techniques coming from the field of *case-based reasoning*. While these approaches rely on heuristic and psychologically inspired formalisms, *terminological logics* have emerged as a powerful representation formalism with a clearly defined formal semantics.

This paper tries to relate case-based reasoning with terminological logics. It demonstrates how terminological logics can serve as a powerful formalism on which the indexing of case bases can be grounded. Using an example from the reuse of plans, it is shown how terminological logics are incorporated into a hybrid representation formalism for plan libraries. A semantics-based indexing scheme is developed and retrieval and update operations are formalized.

3.1 Introduction

Reasoning from second principles has emerged as a new research paradigm in problem solving. Instead of trying to find a problem solution by reasoning from scratch, this method bases the entire problem solving process on the reuse and modification of previous solutions.

Current approaches are settled within the field of *case-based reasoning* which is defined as being a general paradigm for reasoning from experience [Sla89]. Approaches to case-based reasoning mainly rely on psychological theories of human cognition and led to a wide variety of proposals for the representation, indexing, and organization of case bases like *scripts* [SA77] and *dynamic memory* [Sch82].

While case-based reasoning aims at developing a scientific model of human memory, research in *knowledge representation and reasoning* resulted in concept languages of the KL-ONE [Bra78] family, also called *terminological logics*. Terminological logics support a structured representation of abstract knowledge. In contrast to earlier representation formalisms, terminological logics possess a formal semantics. The Tarsky style declarative semantics leads them to be considered as sublanguages of predicate logic [BL84]. With that, the meaning of expressions within the formalism is clearly defined and whether or not the knowledge representation system correctly implements the intended behaviour can be verified. These properties are also desirable for the representation of case bases.

The work described in this paper was motivated by research in *plan reuse*. The reuse and modification of plans is hypothesized as being a valuable tool for improving the efficiency of planning by avoiding the repetition of planning effort. Therefore, plans which have been obtained as solutions for planning problems are stored in *plan libraries* for a further use. The

[1] Project PHI, DFKI Saarbrücken, email: koehler@dfki.uni-sb.de

retrieval of a good plan from a plan library was identified as being a major obstacle for plan reuse systems in a theoretical and empirical analysis of plan reuse versus plan generation [NK93a; NK93b]. Consequently, efficient and theoretically well-founded retrieval and update procedures for plan libraries have to be developed.

The approach which is presented in this paper suggests integrating terminological logics into a hybrid representation formalism for case-based reasoning. Retrieval and update of case bases are grounded on a clearly defined formalism with proper semantics. Their behaviour becomes predictable and formal properties like the completeness and soundness of the retrieval algorithm can be shown.

3.2 A Semantics-Based Indexing of Case Bases

Case-based Reasoning starts with a problem specification for which a solution has to be found in the case base.

Given: a problem specification P_{new}
Wanted: a solution S_{old} for this problem found in a library

To find this solution, a *search key* has to be derived from the problem specification. This search key has to reflect the main properties of the problem at hand. These main properties are often called *deep features*. With the help of deep features, a reusable case is identified in the case base.

In this paper, a completely different view is taken. Instead of searching the case base directly for solutions which can be reused, an indirect search is proposed. It is based on the following observation: the solution which has to be identified in the case base is the result of a previous problem solving process, i.e. S_{old} solves another problem P_{old}.

Observation: S_{old} is the solution to a previously solved problem P_{old}

This suggests that the case base should be searched for previous problem specifications instead of previous solutions. The aim of the search in the case base is the retrieval of an old problem of which the current problem is an instance. If the instance relationship between old and new problem holds, it is sufficient to solve the old problem to find a solution for the new problem.

Question: Given a current problem P_{new}, how to find an appropriate P_{old} the solution of which is sufficient for P_{new}?

When problems are specified in a logical formalism where problem specifications are given as logical formulae, the relation between the current problem and the reused problem can be formally expressed as

$$P_{old} \rightarrow P_{new}$$

This means, that the case base has to be searched for a formal problem specification which logically implies the current one. Furthermore, the case base should be structured based on the implication relation in order to facilitate the identification of a reusable problem. Unfortunately, the implication relation is undecidable in expressive formalisms like first order predicate logic. But the search criterion underlying the retrieval process should at least be decidable. Therefore, instead of using the implication relation directly, an approximation of $P_{old} \rightarrow P_{new}$ is developed. This is done with the help of an *encoding scheme* ω:

$$\boxed{\omega(P_{old}) \rightarrow^{\omega} \omega(P_{new})}$$

The encoding scheme ω has to possess the following formal property:

$$P_{old} \rightarrow P_{new} \Rightarrow \omega(P_{old}) \rightarrow^{\omega} \omega(P_{new})$$

This theorem gives a *monotonicity property* of ω with respect to the extension of the class of models, i.e. if $MOD[P_{old}] \subseteq MOD[P_{new}]$ then $MOD[\omega(P_{old})] \subseteq MOD[\omega(P_{new})]$.

The monotonicity property of the encoding scheme ensures that an existing solution can be found by searching the case base along the $\rightarrow^{\omega}$ dimension. $\omega(P_{old})$ is called the *index* of P_{old} and $\rightarrow^{\omega}$ is defined as a reflexive, and transitive relation between indices. With the help of the encoding scheme a semantics-based similarity measure is obtained: two problems can be defined as being similar, if their indices are related via $\rightarrow^{\omega}$.

Based on the encoding scheme, the *retrieval* of a solution from a case base is formalized as follows:

1. A current problem specification P_{new} is forwarded to the case-based reasoner.
2. The case base contains previous problem specifications P_{old}, the corresponding solutions S_{old} and information extracted from the problem solving process. These entities set up a *case*.
3. Each case possesses an index $\omega(P_{old})$ which determines its position in the hierarchical organized case base.
4. The retrieval process starts with the construction of $\omega(P_{new})$ which represents the index of the current problem. The index is taken as the search key.
5. To identify reusable solutions, all $\omega(P_{old})$ with $\omega(P_{old}) \rightarrow^{\omega} \omega(P_{new})$ are searched.

The *update* of the case base is activated when no reusable case was found. In this situation, the problem P_{new} has to be solved from scratch. After the successful completion of the problem solving process, a solution S_{new} is available. The update procedure builds a new case out of P_{new}, S_{new} and information from the problem solving process. This new case is added to the case base. It is related with its index $\omega(P_{new})$ which determines its position in the case base.

In this paper, terminological logics are proposed as the underlying representation formalism for indices: an index $\omega(P)$ is represented as a *concept* in a terminological logic. The implication between indices $\rightarrow^{\omega}$ is based on the *subsumption relation* between concepts. The retrieval procedure is based on *concept classification*.

To illustrate the definition of a concrete encoding scheme ω for a given application, the representation of a *plan library* for a deductive planning system is considered as an example in the following section.

3.3 Example: Organizing a Plan Library

The PHI System [BBD+93] is a logic-based tool for intelligent help systems. It integrates plan generation as well as plan recognition. Plan generation and plan recognition rely on

the same logical basis which is the interval-based modal temporal logic LLP [BD93]. Plan generation can be done on first principles by planning from scratch and on second principles by reusing previously generated plans [BDK92]. The application domain of PHI is the UNIX *mail domain* where objects like *messages* and *mailboxes* are manipulated by actions like *read*, *delete*, and *save*.

The reuse of plans relies on a four-phase model which provides an abstract formalization of the reuse process [Koe92]. The first phase comprises the retrieval of an appropriate plan from the plan library. The second and third phase are devoted to plan modification [Koe93a]. The fourth phase comprises the dynamic update of the plan library.

Plan modification is formalized as an attempt to prove that the current planning problem is a logical instance of the reused planning problem. Planning problems are represented as logical plan specification formulae of form $[\mathit{preconditions} \wedge \mathsf{Plan}] \rightarrow \mathit{goals}$, i.e., if Plan is carried out in a situation where the *preconditions* hold then the *goals* will be reached.

During the attempt to prove the relation $P_{old} \rightarrow P_{new}$ it is shown that

1. the current preconditions specified in P_{new} are sufficient for the preconditions which the plan S_{old} requires, i.e. by proving

$$\boxed{pre_{new} \rightarrow pre_{old}} \tag{3.1}$$

   it is shown that the reused plan is applicable in the current initial state.

2. the goals specified in P_{old} are sufficient for the current goals required in P_{new}, i.e. by proving

$$\boxed{goal_{old} \rightarrow goal_{new}} \tag{3.2}$$

   it is shown that the reused plan achieves at least all of the current goals.

If both proofs succeed, the plan S_{old} provides a solution for the current planning problem P_{new}. The indexing of the case base should approximate both subproofs, i.e. according to the reasoning level a counterpart on the representational level has to be developed. This is achieved by constructing indices from P_{new} and P_{old} which are represented as concept descriptions in a terminological logic. Indices are added as new representational primitives to the terminological logic and the subsumption relation is extended to them.

3.3.1 Indices as new Representational Primitives

The representation of the plan library is grounded on a hybrid representation formalism integrating the planning logic LLP with a subset of the terminological logic $\mathcal{ALC}$ [SSS91]: A case in this plan library is denoted as a *plan entry*. The information which is stored in each plan entry is represented in the logic LLP. The index of a plan entry is represented as a pair of *admissible concepts* in the terminological logic $\mathcal{ALC}$ [SSS91].

The figure 3.1 illustrates the hybrid representation formalism based on LLP and $\mathcal{ALC}$. The terminological part supports the structuring of the plan library. Retrieval and update are grounded on *classification* by computing the subsumption hierarchy of indices. The planning logic supports the representation of planning knowledge in plan entries.

The terminological logic $\mathcal{ALC}$ was chosen as a starting point for the terminological part of the representation formalism because of its expressiveness and mathematical properties. Concept

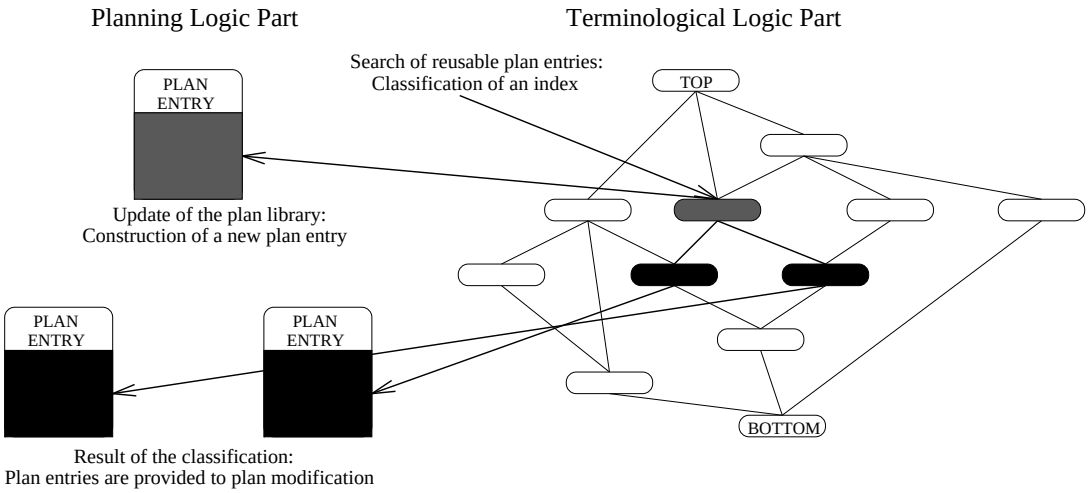

FIGURE 3.1. Hybrid Representation of a Plan Library

descriptions in $\mathcal{ALC}$ are built from concepts, intersection, complements and universal role quantifications. The logic possesses a decidable and complete subsumption algorithm which is PSPACE-complete. In order to obtain a more efficient subsumption algorithm, a subset of $\mathcal{ALC}$ is defined comprising *admissible concepts*. Admissible concepts are consistent concept descriptions in conjunctive normal form which are only built from primitive concepts. The subsumption algorithm for admissible concepts is based on rules for reasoning about lattices [GM92]. It has been shown to be sound, complete and polynomial in [Koe93b].

Indices are represented in $\mathcal{ALC}$ by adding new representational primitives to the logic, an idea which was taken from the system RAT [JHP93]. An index is a tuple $\langle \omega(pre), \omega(goal) \rangle$ where $\omega(pre), \omega(goal)$ are admissible concepts.

The subsumption relation between admissible concepts is extended to indices using the so-called *applicability subsumption* developed in RAT. Thereby subsumption between indices can be reduced to subsumption between preconditions and goals.

$$\begin{aligned} \omega(P_{old}) \rightarrow^{\omega} \omega(P_{new}) \quad &\leftrightarrow \quad \langle \omega(pre_{old}), \omega(goal_{old}) \rangle \sqsubseteq \langle \omega(pre_{new}), \omega(goal_{new}) \rangle \\ &\leftrightarrow \quad \omega(pre_{new}) \sqsubseteq \omega(pre_{old}) \wedge \omega(goal_{old}) \sqsubseteq \omega(goal_{new}) \end{aligned} \tag{3.3}$$

Comparing 3.1 and 3.2 with 3.3 it can be observed that $\rightarrow^{\omega}$ as defined above leads to the required representational counterpart.

3.3.2 The Encoding Scheme ω

The encoding scheme ω maps LLP plan specifications to indices in $\mathcal{ALC}$. It depends on the source logic LLP as well as on the target logic $\mathcal{ALC}$.

The encoding starts with an *abstraction* and *normalization* of LLP specification formulae. During the abstraction process, an LLP formula is replaced by a weaker logical formula applying first order and modal abstraction rules. These rules abstract from the ordering of intermediate subgoal states which can be specified in LLP and from concrete objects occurring in plan specifications.

During the normalization, the formulae specifying preconditions and goals are transformed into conjunctive normal form. Furthermore, modal formulae are translated into first order formulae according to an approach developed in [KT93]. Finally, information represented in first order terms is encoded with the help of predicates. The process is rather complex and cannot described in detail here because of space restrictions; the reader is referred to [Koe93b].

The abstraction and normalization process leads to a unified representation of plan specifications which can be directly transformed into admissible concepts. In this unified representation, preconditions and goals are specified by formulae in conjunctive normal form which contain only existentially quantified binary predicates.

The encoding scheme maps first order binary predicates to *roles* in $\mathcal{ALC}$. Equations occurring in normalized specification formulae are encoded as *existential role restrictions*. Variables are encoded as *primitive concepts*. The result of the encoding are the admissible concepts $\omega(pre)$ and $\omega(goal)$ from which the index of the plan specification is obtained.

3.4 Conclusion

An approach for developing a semantics-based indexing of case bases using terminological logics has been presented. Indices are built from concept descriptions. These concept descriptions are constructed from formal problem specification with the help of an encoding scheme. Since indices are added to terminological logics as new representational primitives, the subsumption relation defined for concept descriptions can be extended to indices. The retrieval and update operations working on case bases can be formalized as classification operations over the taxonomy of indices. Due to the formalization they are assigned a clearly defined semantics. Thereby, the behavior of the operations becomes predictable and the theoretical properties can be proved.

3.5 Acknowledgement

I wish to thank Hans-Jürgen Profitlich who helped me test the practical feasibility of the approach with the development of a prototypical plan library in RAT and Bernhard Nebel for his advice regarding the theoretical properties of the formalism.

3.6 References

[BBD+93] M. Bauer, S. Biundo, D. Dengler, J.Koehler, and G. Paul. PHI - a logic-based tool for intelligent help systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993. Morgan Kaufmann, Menlo Park.

[BD93] S. Biundo and D. Dengler. The logical language for planning LLP. DFKI Research Report, German Research Center for Artificial Intelligence, 1993. to appear.

[BDK92] S. Biundo, D. Dengler, and J. Koehler. Deductive planning and plan reuse in a command language environment. In *Proceedings of the 10th European Conference*

on Artificial Intelligence, pages 628–632, 1992.

[BL84] R. Brachmann and H. Levesque. The tractability of subsumption in frame based description languages. In *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84)*, pages 34–37, 1984.

[Bra78] R. Brachman. Structured inheritance networks. In W. Woods and R. Brachman, editors, *Research in Natural Language Understanding*, pages 36–78. Bolt, Beranek, and Newman Inc., Cambridge Mass., 1978.

[GM92] R. Givan and D. McAllester. New results on local inference relations. In B. Nebel et al., editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, pages 403–412. Morgan Kaufman, 1992.

[JHP93] B. Nebel J. Heinsohn, D. Kudenko and H.-J. Profitlich. RAT: Representation of actions using terminological logics. DFKI Research Report, German Research Center for Artificial Intelligence, 1993. to appear.

[Koe92] J. Koehler. Towards a logical treatment of plan reuse. In *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, pages 285–286, Washington, D.C., 1992. Morgan Kaufmann, Menlo Park.

[Koe93a] J. Koehler. Flexible plan reuse in a formal framework. DFKI Research Report, German Research Center for Artificial Intelligence, 1993. to appear.

[Koe93b] J. Koehler. Representing plan libraries. DFKI Research Report, German Research Center for Artificial Intelligence, 1993. to appear.

[KT93] J. Koehler and R. Treinen. Constraint deduction in an interval-based temporal logic. In *Working Notes of the AAAI Symposium on Automated Deduction in Nonstandard Logics*. AAAI Press, Menlo Park, 1993.

[NK93a] B. Nebel and J. Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993. Morgan Kaufmann, Menlo Park.

[NK93b] B. Nebel and J. Koehler. Plan reuse versus plan generation: A theoretical and empirical analysis. DFKI Research Report RR-93-33, German Research Center for Artificial Intelligence, 1993.

[SA77] R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Earlbaum, Hillsdale, N.J., 1977.

[Sch82] R. Schank. *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, Cambridge, 1982.

[Sla89] S. Slade. Case-based reasoning: A research paradigm. *The AI Magazine*, 12(1), 1989.

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

4

A Terminological Knowledge Representation System with Complete Inference Algorithms

Franz Baader[1]
Bernhard Hollunder[2]

ABSTRACT The knowledge representation system KL-ONE first appeared in 1977. Since then many systems based on the idea of KL-ONE have been built. The formal model-theoretic semantics which has been introduced for KL-ONE languages [BL84] provides means for investigating soundness and completeness of inference algorithms. It turned out that almost all implemented KL-ONE systems such as BACK, KL-TWO, LOOM, NIKL, SB-ONE use sound but incomplete algorithms.

Until recently, sound *and* complete algorithms for the basic reasoning facilities in these systems such as consistency checking, subsumption checking (classification) and realization were only known for rather trivial languages. However, in the last two years concept languages (term subsumption languages) have been thoroughly investigated (see for example [SS88; Neb90; HNS90; DLNN91]). As a result of these investigations it is now possible to provide sound and complete algorithms for relatively large concept languages.

In this paper we describe $\mathcal{KRIS}$ which is an implemented prototype of a KL-ONE system where all reasoning facilities are realized by sound and complete algorithms. This system can be used to investigate the behaviour of sound and complete algorithms in practical applications. Hopefully, this may shed a new light on the usefulness of complete algorithms for practical applications, even if their worst case complexity is NP or worse.

$\mathcal{KRIS}$ provides a very expressive concept language, an assertional language, and sound and complete algorithms for reasoning. We have chosen the concept language such that it contains most of the constructs used in KL-ONE systems with the obvious restriction that the interesting inferences such as consistency checking, subsumption checking, and realization are decidable. The assertional language is similar to languages normally used in such systems. The reasoning component of $\mathcal{KRIS}$ depends on sound and complete algorithms for reasoning facilities such as consistency checking, subsumption checking, retrieval, and querying.

4.1 Introduction and Motivation

In the last decade many knowledge representation systems in the tradition of KL-ONE [BS85] have been built, for example BACK [NvL88; Neb90], CLASSIC [BBMR89], KANDOR [Pat84], KL-TWO [Vil85], KRYPTON [BPGL85], LOOM [MB87], NIKL [KBR86], SB-ONE [Kob89]. A common feature of these systems is the separation of the knowledge into a terminological part and an assertional part. Knowledge about classes of individuals and relationships between these classes is stored in the *TBox*, and knowledge concerning particular individuals can be described in the *ABox*.

The TBox formalism provides a *concept language* (or term subsumption language) for the definition of concepts and roles, where concepts are interpreted as sets of individuals and

[1]Project TACOS, DFKI Saarbrücken, email: baader@dfki.uni-sb.de
[2]Project TACOS, DFKI Saarbrücken, email: beho@dfki.uni-sb.de

roles as binary relations between individuals. Starting with primitive concepts and roles the language formalism is used to build up more complex concepts and roles.

For example, assume that `person`, `female`, and `shy` are primitive concepts, and `child` and `female_relative` are primitive roles. Taking the connectives concept conjunction (and), disjunction (or), and negation (not) one can express "persons who are female or not shy" by

(and person (or female (not shy))).

Since concepts are interpreted as sets, concept conjunction can be interpreted as set intersection, concept disjunction as set union, and negation of concepts as set complement. In addition to these operations on sets one can also employ roles for the definition of new concepts. *Value restrictions* can be used for instance to describe "individuals for whom all children are female" by the expression (all `child` `female`). *Number restrictions* allow for instance to describe "individuals having at most three children" by the expression (atmost 3 `child`). Beside the above mentioned constructs there are other well-known concept-forming constructs which are available in $\mathcal{KRIS}$ (see Section 2). An example for a role-forming construct is the conjunction of roles. We can define the role (and `child` `female_relative`), which intuitively yields the role daughter. The concept language presented in the next section also provides functional roles, so-called *attributes*. These attributes are interpreted as partial functions and not as arbitrary binary relations. Natural examples for attributes may be `father` or `first_name`. An *agreement* between two attribute chains for example allows to describe "individuals whose father and grandfather have the same first name" by the expression

(equal (compose `father` `first_name`) (compose `father` `father` `first_name`)).

Interestingly, agreements between attribute chains do not make reasoning in the language undecidable [HN90], whereas agreements between arbitrary role chains cause undecidability [Sch89].

The basic reasoning facilities concerning the TBox are the determination whether a concept denotes nothing, i.e., whether a concept denotes the empty set in every interpretation, and the computation of the subsumption hierarchy. A concept C subsumes (is more general than) a concept D iff in every interpretation the set denoted by C is a superset of the set denoted by D.

The ABox formalism consists of an assertional language which allows the introduction of individuals to express facts about a concrete world. One can state that individuals are instances of concepts, and that pairs of individuals are instances of roles or attributes.

The reasoning facilities concerning both the TBox and the ABox are classified as follows. We need algorithms for inferences such as

- checking the consistency of the represented knowledge,
- given an individual of the ABox, compute the most specific concepts in the TBox this individual is instance of,
- computing all individuals of the ABox that are instances of a given concept.

The formal model-theoretic semantics which has been introduced for KL-ONE languages [BL84] provides means for investigating soundness and completeness of inference algorithms. It turned out that the above mentioned systems use sound but incomplete algorithms. If a sound but incomplete subsumption algorithm detects a subsumption relation, this relation

really exists; but if it fails to recognize that a concept subsumes another one, then we do not know anything. A subsumption relation may or may not exist. Thus, the results of the algorithms only partially coincides with what the formal semantics expresses.[3]

Until recently, sound *and* complete algorithms for the above mentioned inferences and for the subsumption problem were only known for rather trivial languages which explains the use of incomplete algorithms in existing KL-ONE systems. Another argument in favour of incomplete algorithms was that for many languages the subsumption problem is at least NP-hard [LB87; Neb88]. Consequently, complete algorithms have to be intractable, whereas incomplete algorithms may still be polynomial. However, one should keep in mind that these complexity results are worst case results. It is not at all clear how complete algorithms may behave for typical knowledge bases.

In [SS88; HNS90; Hol90] it is shown how to devise sound and complete algorithms for the above mentioned inferences in various concept languages. Thus it has become possible to implement a KL-ONE system ($\mathcal{KRIS}$) which provides

- a very expressive concept language,
- powerful reasoning facilities, and
- sound and complete algorithms for these facilities.

The purpose of this paper is as follows. Firstly, we will enumerate the language constructs which are available in $\mathcal{KRIS}$, and will give a formal semantics for their meaning. We have chosen the concept language such that it contains most of the constructs used in KL-ONE systems with the obvious restriction that the interesting inferences such as consistency checking, subsumption checking, and realization are decidable. Of course, taking such a large language means that the complexity of the inference algorithms is relatively high. But $\mathcal{KRIS}$ also provides faster algorithms for certain sublanguages.[4] Secondly, we will describe the inference mechanisms provided by $\mathcal{KRIS}$.

4.2 Formalisms for Representing Knowledge

In this section we will introduce the formalisms for representing knowledge in $\mathcal{KRIS}$. In Subsection 4.2.1 the syntax and semantics of the concept language and the terminological axioms are presented. In Subsection 4.2.2 the assertional language and its semantics are introduced.

4.2.1 The Concept Language Underlying $\mathcal{KRIS}$

Assume that we have three disjoint alphabets of symbols, called *concept names*, *role names*, and *attribute names*. The special concept name $*\mathsf{top}*$ is called *top concept*.

[3] But see Patel-Schneider [Pat89] who uses a four-valued semantics to formally describe the behaviour of an algorithm which is incomplete w.r.t. two-valued semantics.

[4] That coincides with what Ramesh Patil proposed at the Workshop on Term Subsumption Languages in Knowledge Representation: "He therefore strongly opposed any attempt to further restrict the expressiveness of TSL (term subsumption language) systems. Instead, he proposed that such systems be configured on a "pay as you go" basis—if the application uses only a small portion of the expressive power of the TSL, then everything will be fast; if more expressive power is used, then the system may slow down, but still be able to represent and reason with the knowledge given to it." (see [PSOK+90]).

The sets of *concept terms*, *role terms*, and *attribute terms* are inductively defined as follows. Every concept name is a concept term, every role name is a role term, and every attribute name is an attribute term. Now let $C, C_1, \ldots, C_k$ be concept terms, $R, R_1, \ldots, R_l$ be role terms, $f, g, f_1, \ldots, f_m$ be attribute terms already defined, and let n be a nonnegative integer. Then

(and $C_1 \ldots C_k$),	(conjunction)
(or $C_1 \ldots C_k$),	(disjunction)
(not C),	(negation)
(all R C), (all f C),	(value restriction)
(some R C), (some f C),	(exists restriction)
(atleast n R) (atmost n R)	(number restrictions)
(equal f g),	(agreement)
(not-equal f g)	(disagreement)

are concept terms,

(and $R_1 \ldots R_l$)	(role conjunction)

is a role term, and

(compose $f_1 \ldots f_m$)	(composition)

is an attribute term.

So-called *terminological axioms* are used to introduce names for concept, role, and attribute terms. A finite set of such axioms satisfying certain restrictions is called a terminology (TBox). There are three different ways of introducing new concepts (respectively roles or attributes) into a terminology.

Let A (P, f) be a concept (role, attribute) name, and let C (R, g) be a concept (role, attribute) term. By the terminological axioms

$$(\textsf{defprimconcept } A), \quad (\textsf{defprimrole } P), \quad (\textsf{defprimattribute } f)$$

new concept, role, and attribute names are introduced without restricting their interpretation. The terminological axioms

$$(\textsf{defprimconcept } A\ C), \quad (\textsf{defprimrole } P\ R), \quad (\textsf{defprimattribute } f\ g)$$

impose necessary conditions on the interpretation of the introduced concept, role, and attribute names. Finally, one can impose necessary and sufficient conditions by the terminological axioms

$$(\textsf{defconcept } A\ C), \quad (\textsf{defrole } P\ R), \quad (\textsf{defattribute } f\ g).$$

A *terminology* (*TBox*) $\mathcal{T}$ is a finite set of terminological axioms with the additional restriction that (*i*) every concept, role, and attribute name may appear at most once as a first argument of a terminological axiom in $\mathcal{T}$ (unique definition), and (*ii*) $\mathcal{T}$ must not contain cyclic definitions[5] (acyclicity).

A terminology which describes knowledge about persons and relationships between persons is shown in Figure 4.1. At first, the attribute `sex` and the concept `male` is introduced. The

[5] For a discussion of terminological cycles see [Neb88; Baa90a].

```
(defprimattribute sex)
(defprimconcept male)
(defprimconcept female (not male))
(defprimconcept person (some sex (or male female)))
(defprimrole child)
(defconcept parent (and person (some child person)))
(defconcept mother (and parent (some sex female)))
(defconcept father (and parent (not mother)))
(defconcept grandparent (and parent (some child parent)))
(defconcept parent_with_two_children (and parent (atleast 2 child)))
(defconcept parent_with_sons_only (and parent (all child (some sex male))))
```

FIGURE 4.1. A terminology (TBox).

axioms which define the concepts `female` and `person` can be read as follows: "no individual is both male and female"[6], and "a person has sex male or female." These axioms impose necessary conditions on the interpretation of the introduced concepts. The definition of the concept `parent` impose necessary and sufficient conditions: "an individual is a parent if and only if it is a person and has some child who is a person." The other concepts are also defined according to their intuitive meaning.

We will now give a formal model-theoretic semantics for the concept language and the terminological axioms. An *interpretation* $\mathcal{I}$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$). The interpretation function maps every concept name A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, every role name P to a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every attribute name f to a partial function $f^{\mathcal{I}}$ from $\Delta^{\mathcal{I}}$ to $\Delta^{\mathcal{I}}$. With $dom\, f^{\mathcal{I}}$ we denote the domain of the partial function $f^{\mathcal{I}}$ (i.e., the set of elements of $\Delta^{\mathcal{I}}$ for which $f^{\mathcal{I}}$ is defined).

The interpretation function—which gives an interpretation for concept, role, and attribute names—can be extended to concept, role, and attribute terms as follows. Let $C, C_1, \ldots, C_k$ be concept terms, $R, R_1, \ldots, R_l$ role terms, $f, g, f_1, \ldots, f_m$ attribute terms, and let n be a nonnegative integer. Assume that $C^{\mathcal{I}}$, $C_1^{\mathcal{I}}, \ldots, C_k^{\mathcal{I}}$, $R^{\mathcal{I}}$, $R_1^{\mathcal{I}}, \ldots, R_l^{\mathcal{I}}$, $f^{\mathcal{I}}$, $g^{\mathcal{I}}$, $f_1^{\mathcal{I}}, \ldots, f_m^{\mathcal{I}}$ are already defined. Then

$$
\begin{aligned}
(*\mathsf{top}*)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \\
(\mathsf{and}\ C_1 \ldots C_k)^{\mathcal{I}} &:= C_1^{\mathcal{I}} \cap \ldots \cap C_k^{\mathcal{I}} \\
(\mathsf{or}\ C_1 \ldots C_k)^{\mathcal{I}} &:= C_1^{\mathcal{I}} \cup \ldots \cup C_k^{\mathcal{I}} \\
(\mathsf{not}\ C)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\mathsf{all}\ R\ C)^{\mathcal{I}} &:= \{\, a \in \Delta^{\mathcal{I}} \mid \forall b : (a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}} \} \\
(\mathsf{all}\ f\ C)^{\mathcal{I}} &:= \{\, a \in \Delta^{\mathcal{I}} \mid a \in dom\, f^{\mathcal{I}} \Rightarrow f^{\mathcal{I}}(a) \in C^{\mathcal{I}} \} \\
(\mathsf{some}\ R\ C)^{\mathcal{I}} &:= \{\, a \in \Delta^{\mathcal{I}} \mid \exists b : (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \} \\
(\mathsf{some}\ f\ C)^{\mathcal{I}} &:= \{\, a \in dom\, f^{\mathcal{I}} \mid f^{\mathcal{I}}(a) \in C^{\mathcal{I}} \} \\
(\mathsf{atleast}\ n\ R)^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}| \geq n\} \\
(\mathsf{atmost}\ n\ R)^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}| \leq n\} \\
(\mathsf{equal}\ f\ g)^{\mathcal{I}} &:= \{\, a \in dom\, f^{\mathcal{I}} \cap dom\, g^{\mathcal{I}} \mid f^{\mathcal{I}}(a) = g^{\mathcal{I}}(a) \} \\
(\mathsf{not\text{-}equal}\ f\ g)^{\mathcal{I}} &:= \{\, a \in dom\, f^{\mathcal{I}} \cap dom\, g^{\mathcal{I}} \mid f^{\mathcal{I}}(a) \neq g^{\mathcal{I}}(a) \} \\
(\mathsf{and}\ R_1 \ldots R_l)^{\mathcal{I}} &:= R_1^{\mathcal{I}} \cap \ldots \cap R_l^{\mathcal{I}} \\
(\mathsf{compose}\ f_1 \ldots f_m)^{\mathcal{I}} &:= f_1^{\mathcal{I}} \circ \ldots \circ f_m^{\mathcal{I}},
\end{aligned}
$$

[6] It might seem to be more convenient to allow explicit disjointness axioms for expressing such facts. In fact, we could easily provide such axioms at the user interface because they can be simulated by the constructs available in our language [Neb90; Baa90b].

where $|X|$ denotes the cardinality of the set X and $\circ$ denotes the composition of functions. The composition should be read from left to right, i.e., $f_1^{\mathcal{I}} \circ \ldots \circ f_m^{\mathcal{I}}$ means that $f_1^{\mathcal{I}}$ is applied first, then $f_2^{\mathcal{I}}$, and so on. Note, that if $f_1^{\mathcal{I}}, \ldots, f_m^{\mathcal{I}}$ are partial functions, then $f_1^{\mathcal{I}} \circ \ldots \circ f_m^{\mathcal{I}}$ is also a partial function.

The semantics of terminological axioms is now defined as follows. An interpretation $\mathcal{I}$ *satisfies* the terminological axiom

(defprimconcept A C)	iff	$A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$,
(defconcept A C)	iff	$A^{\mathcal{I}} = C^{\mathcal{I}}$,
(defprimrole P R)	iff	$P^{\mathcal{I}} \subseteq R^{\mathcal{I}}$,
(defrole P R)	iff	$P^{\mathcal{I}} = R^{\mathcal{I}}$,
(defprimattribute f g)	iff	$f^{\mathcal{I}} \subseteq g^{\mathcal{I}}$,
(defattribute f g)	iff	$f^{\mathcal{I}} = g^{\mathcal{I}}$,

where A (P, f) is a concept (role, attribute) name, and C (R, g) is a concept (role, attribute) term. Note that the terminological axioms (defprimconcept A), (defprimrole P), and (defprimattribute f) are satisfied in every interpretation by the definition of interpretation. An interpretation $\mathcal{I}$ is a *model* for a TBox $\mathcal{T}$ iff $\mathcal{I}$ satisfies all terminological axioms in $\mathcal{T}$.

4.2.2 Assertions

The assertional formalism allows to introduce individuals (objects). We can describe a concrete world by stating that individuals are instances of concepts, and that pairs of individuals are instances of roles or attributes.

Assume that we have a further alphabet of symbols, called *individual names*. Names for individuals are introduced by *assertional axioms* which have the form

$$\text{(assert-ind } a\ C\text{)}, \quad \text{(assert-ind } a\ b\ R\text{)}, \quad \text{(assert-ind } a\ b\ g\text{)},$$

where a, b are individual names, and C (R, g) is a concept (role, attribute) term. A *world description* (*ABox*) is a finite set of assertional axioms.

Figure 4.2 shows an example of an ABox. This ABox describes a world in which Tom is father

```
(assert-ind Tom father)
(assert-ind Tom Peter child)                (assert-ind Tom Harry child)
(assert-ind Mary parent_with_sons_only)
(assert-ind Mary Tom child)                 (assert-ind Mary Chris child)
```

FIGURE 4.2. A world description (ABox).

of Peter and Harry. Furthermore, Mary has only sons; two of them are Tom and Chris.

Note that an ABox can be considered as a relational database where the arity of each tuple is either one or two. However, in contrast to the closed world semantics which is usually employed in databases, we assume an *open world semantics*, since we want to allow for incomplete knowledge. Thus, we cannot conclude in the above example that Tom has exactly two children, since there may exist a world in which Tom has some additional children.

The semantics of individual names and assertional axioms is defined as follows. The interpretation function $\cdot^{\mathcal{I}}$ of a TBox interpretation $\mathcal{I}$ can be extended to individual names by mapping them to elements of the domain such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$. This restriction on the interpretation function ensures that individuals with different names denote different

individuals in the world. It is called *unique name assumption*, which is usually also assumed in the database world.

Let a, b be individual names, and C (R, g) be a concept (role, attribute) term. An interpretation $\mathcal{I}$ *satisfies* the assertional axiom

$$\begin{array}{lll} (\textsf{assert-ind } a\ C) & \text{iff} & a^{\mathcal{I}} \in C^{\mathcal{I}} \\ (\textsf{assert-ind } a\ b\ R) & \text{iff} & (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}} \\ (\textsf{assert-ind } a\ b\ f) & \text{iff} & f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}}. \end{array}$$

The semantics of an ABox together with a TBox is defined as follows. We say that an interpretation $\mathcal{I}$ is a *model* for an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies all assertional axioms in $\mathcal{A}$ and all terminological axioms in $\mathcal{T}$.

4.3 Reasoning

In this section we describe the inference mechanisms provided by $\mathcal{KRIS}$. The reasoning component of $\mathcal{KRIS}$ allows to make knowledge explicit which is only implicitly represented in an ABox and a TBox. For example, from the TBox and Abox given in the previous section one can conclude that `Mary` is a grandparent, though this knowledge is not explicitly stored in the ABox.

An obvious requirement on the represented knowledge is that it should be consistent since everything would be deducible from inconsistent knowledge (from a logical point of view). If, for instance, an ABox contains the axioms (assert-ind `Chris mother`), (assert-ind `Chris father`), then the system should detect this inconsistency.[7] The underlying model-theoretic semantics allows a clear and intuitive definition of consistency. We say that an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ is *consistent* if it has a model. Thus, we have the

Consistency problem: Does there exist a model for $\mathcal{A}$ w.r.t. $\mathcal{T}$?

Beside an algorithm for checking consistency of an ABox w.r.t. a TBox $\mathcal{KRIS}$ provides algorithms for the basic reasoning facilities such as subsumption and instantiation.

We say that a concept term C *subsumes* a concept term D iff $C^{\mathcal{I}} \supseteq D^{\mathcal{I}}$ in every interpretation $\mathcal{I}$. Thus, an algorithm for checking subsumption takes concept terms C and D as arguments and has to solve the

Subsumption problem: Does C subsume D ?

The subsumption problem in concept languages has been thoroughly investigated in [SS88; HNS90; DLNN91]. In these papers subsumption algorithms for various concept languages and sublanguages are given and their computational complexity is discussed. In fact, the papers do not directly describe subsumption algorithms but algorithms for a closely related problem. These algorithms check whether a given concept term C is meaningful, i.e., whether there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$. Since C subsumes D if and only if $(\textsf{and } D\ (\textsf{not } C))^{\mathcal{I}} = \emptyset$ for every interpretation $\mathcal{I}$, these algorithms can also be used to decide subsumption.

[7] However, in general it is not always as easy as in this example to check whether the represented knowledge is consistent.

An algorithm for instantiation decides whether an assertional axiom is deducible from the represented knowledge. More formally, let α be an assertional axiom. We say that an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ *implies* α iff all models for $\mathcal{A}$ w.r.t. $\mathcal{T}$ satisfy α, written $\mathcal{A}, \mathcal{T} \models \alpha$. Thus we define the

Instantiation problem: Is α implied by $\mathcal{A}$ and $\mathcal{T}$?

If α is of the form (assert-ind a b R) or (assert-ind a b f), then it is relatively easy to solve the instantiation problem since the concept language allows only few constructs to build complex role or attribute terms. If α is of the form (assert-ind a C), the instantiation problem can be reduced to the consistency problem as follows:

$$\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a\ C)$$
$$\text{iff}$$
$$\mathcal{A} \cup \{(\text{assert-ind } a\ (\text{not } C))\} \text{ w.r.t. } \mathcal{T} \text{ is not consistent.}$$

In [Hol90] a sound and complete algorithm for the consistency and instantiation problem for a sublanguage of the language defined in Section 4.2 is described.

$\mathcal{KRIS}$ also provides algorithms which find out certain relationships between the defined concepts, roles, attributes, and individuals. These algorithms are based on the algorithms for subsumption and instantiation. Assume that $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox.

The *subsumption hierarchy* is the preordering of the concept names in $\mathcal{T}$ w.r.t. the subsumption relation. The so-called *classifier* has to solve the

Classification problem: Compute the subsumption hierarchy.

Given an individual in $\mathcal{A}$, one wants to know the set of concept names in $\mathcal{T}$ which describe it most accurately. To be more formal, let a be an individual occurring in $\mathcal{A}$. The set of *most specialized concepts* of a is a set $\{A_1, \ldots, A_n\}$ of concept names occurring in $\mathcal{T}$ such that

1. $\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a\ A_i)$ for every i, $1 \leq i \leq n$,
2. for every i, $1 \leq i \leq n$, there does not exist a concept name A in $\mathcal{T}$ such that $\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a\ A)$, A_i subsumes A, and A does not subsume A_i, and
3. for every concept name A in $\mathcal{T}$ such that $\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a\ A)$, there exists an A_i such that A subsumes A_i.

The first condition means that each A_i is in fact a description of a. The second condition guarantees that the set contains only the smallest description w.r.t. the subsumption relation, and the third condition means that we do not omit any nonredundant description. Thus, to describe an individual most accurately we need an algorithm for the

Realization problem: Compute for an individual in $\mathcal{A}$ the set of most specialized concepts in $\mathcal{T}$.

Conversely, we want to know the individuals of $\mathcal{A}$ which are instances of a given concept term. Let C be a concept term. The set $INST(C)$ contains all the individuals $a_1, \ldots, a_n$ of $\mathcal{A}$ such that $\mathcal{A}, \mathcal{T} \models (\text{assert-ind } a_i\ C)$ holds. We have the

Retrieval problem: Compute for a given concept term C the set $INST(C)$.

4.4 Summary and Outlook

The $\mathcal{KRIS}$ system which has been presented in this paper distinguishes itself from all the other implemented KL-ONE based systems in that it employs complete inference algorithms. Nevertheless its concept language is relatively large. Of course, the price one has to pay is that the worst case complexity of the algorithms is worse than NP. However, it is not *a priori* clear whether this also implies that $\mathcal{KRIS}$ has to be less efficient for "typical" knowledge bases. An important reason for implementing the $\mathcal{KRIS}$ system was that it could be used to investigate this question. The empirical analysis of terminological representation systems described in [HKNP92] showed that $\mathcal{KRIS}$ was much slower than, for example, CLASSIC, even for knowledge bases that are in the scope of CLASSIC's concept language, and for which CLASSIC's subsumption algorithm is complete. However, in [BFHNP92] it has been shown that this bad performance of $\mathcal{KRIS}$ is not mainly due to the use of complete subsumption algorithms, but instead to the fact that the tested version was the first implementation of an experimental system where efficiency considerations only played a minor role. For this purpose we investigated possible optimizations of the classification process. The optimization techniques that came best off were incorporated in the $\mathcal{KRIS}$ system. These, together with more conventional optimizations on the implementation level, led to a significant speed up so that the new version has now a runtime behavior similar to that of the other systems on the test data used in [HKNP92].

In the current project TACOS we are investigating extensions of terminological knowledge representation with several forms of commonsense knowledge: temporal knowledge and epistemic knowledge in the subproject TACOS-M; incomplete, uncertain, and vague knowledge in the subproject TACOS-P. For the temporal and epistemic part we are focussing on modal logic approaches, i.e., we consider terminological fragments of certain modal predicate logics. The modal operators will be interpreted temporally or as knowledge and belief (cf., e.g., [Sch92; Lau92]) To deal with incomplete knowledge we investigate different nonmonotonic extensions of terminological logics (cf., e.g., [BH92; DLNNS92; BH93]) For the representation of uncertain and vague knowledge we consider possibilistic logic, probabilistic logic, and related non-standard approaches.

Our main research objectives are to provide, if possible, decidable extensions of this kind. More precisely, we want to design sound and complete algorithms for the usual inference services of terminological systems in the extended formalism. The extensions and the corresponding algorithms will be integrated into $\mathcal{KRIS}$.

4.5 References

[Baa90a] F. Baader. "Terminological Cycles in KL-ONE-based Knowledge Representation Languages." In *Proceedings of the 8th National Conference of the AAAI*, pp. 621-626, Boston, Mas., 1990.

[Baa90b] F. Baader. "A Formal Definition for the Expressive Power of Knowledge Representation Languages." In *Proceedings of the 9th European Conference on Artificial Intelligence*, pp. 53–58, Stockholm, Sweden, 1990.

[Baa90c] F. Baader. "Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles." In *Proceedings of IJCAI '91.*

[BBHH+90] F. Baader, H.-J. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel,

W. Nutt, H.-J. Profitlich. *Terminological Knowledge Representation: A Proposal for a Terminological Logic*. DFKI Technical Memo TM-90-04, DFKI, Postfach 2080, D-6750 Kaiserslautern, Germany.

[BBHNS90] F. Baader, H.-J. Bürckert, B. Hollunder, W. Nutt, J. H. Siekmann. "Concept Logics" In *Proceedings of the Symposium on Computational Logics*, Brüssel, November 1990.

[BFHNP92] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. "An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on." In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning*, Cambridge, Mass., 1992.

[BH90] F. Baader, P. Hanschke. "A Schema for Integrating Concrete Domains into Concept Languages." In *Proceedings of IJCAI '91.*

[BH93] F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. Research Report RR-92-58, DFKI Saarbrücken, 1992. Also to appear in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993.

[BH92] F. Baader and B. Hollunder: "Embedding Defaults into Terminological Knowledge Representation Formalisms." In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning, 1992.*

[BBMR89] A. Borgida, R. J. Brachman, D. L. McGuinness, L. A. Resnick. "CLASSIC: A Structural Data Model for Objects." In *Proceedings of the International Conference on Management of Data*, Portland, Oregon, 1989.

[BPGL85] R. J. Brachman, V. Pigman Gilbert, H. J. Levesque. "An essential hybrid reasoning system: knowledge and symbol level accounts in KRYPTON." In *Proceedings of the 9th IJCAI,* pp. 532–539, Los Angeles, Cal., 1985.

[BL84] R. J. Brachmann, H. J. Levesque. "The tractability of subsumption in frame based description languages." In *Proceedings of the 4th National Conference of the AAAI,* pp. 34–37, Austin, Tex., 1984.

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

[Bür90] H.-J. Bürckert. "A Resolution Principle for Clauses with Constraints" In *Proceedings of the 10th International Conference on Automated Deduction*, Lecture Notes in Artificial Intelligence, LNAI 449, Springer Verlag, pp. 178-192, 1990.

[BM91] H.-J. Bürckert, J. Müller. "RATMAN: A Rational Agent Testbed for Multi Agent Networks", In *Proceedings of Modeling Autonomous Agents in Multi-Agent Worlds*, Elsevier Publishers, 1991.

[DLNN91] 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.

[DLNNS92] F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf: "Adding Epistemic Operators to Concept Languages." In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning, 1992.*

[HKNP92] J. Heinsohn, D. Kudenko, B. Nebel, H.-J. Profitlich. "An empirical analysis of terminological representation systems." In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pp. 767–773, San Jose, CA, July 1992. MIT Press.

[Hol90] 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 Description Languages*. DFKI Research Report RR-90-04, DFKI, Postfach 2080, D-6750 Kaiserslautern, Germany.

[HNS90] B. Hollunder, W. Nutt, M. Schmidt-Schauß. "Subsumption Algorithms for Concept Description Languages." In *Proceedings of the 9th European Conference on Artificial Intelligence*, pp. 348–353, Stockholm, Sweden, 1990.

[KBR86] T. S. Kaczmarek, R. Bates, G. Robins. "Recent developments in NIKL." In *Proceedings of the 5th National Conference of the AAAI*, pp. 578–587, Philadelphia, Pa., 1986.

[Kob89] A. Kobsa. "The SB-ONE knowledge representation workbanch" In *Preprints of the Workshop on Formal Aspects of Semantic Networks*, Two Harbors, Cal., February 1989.

[Lau92] A. Laux: *Integrating a Modal Logic of Knowledge into Terminological Logics*, DFKI Research Report RR-92-56, 1992.

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

[MB87] R. MacGregor, R. Bates. *The Loom Knowledge Representation Language*. Technical Report ISI/RS-87-188, University of Southern California, Information Science Institute, Marina del Rey, Cal., 1987.

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

[Neb89] B. Nebel. "Terminological Cycles: Semantics and Computational Properties." In *Proceedings of the Workshop on Formal Aspects of Semantic Networks*, Two Harbors, Cal., February 1989.

[Neb88] B. Nebel. "Computational complexity of terminological reasoning in BACK." *Artificial Intelligence*, 34(3):371–383, 1988.

[NvL88] B. Nebel, K. von Luck. "Hybrid Reasoning in BACK." In Z. W. Ras, L. Saitta (editors), *Methodologies for Intelligent Systems*, pp. 260–269, North Holland, Amsterdam, Netherlands, 1988.

[Pat84] P. Patel-Schneider. "Small can be beautiful in knowledge representation." In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pp. 11–16, Denver, Colo., 1984.

[Pat89] P. Patel-Schneider. "A four-valued Semantics for Terminological Logics." *Artificial Intelligence*, 39(2):263-272, 1989.

[PSOK+90] P. Patel-Schneider, B. Owsnicki-Klewe, A. Kobsa, N. Guarino, R. MacGregor, W. S. Mark, D. L. McGuinness, B. Nebel, A. Schmiedel, J. Yen. "Term Subsumption in Knowledge Representation." In *AI Magazine*, 11(2):16-23, 1990. pp. 11–16, Denver, Colo., 1984.

[Sch92] K. Schild. Combining terminological logics with tense logic. Technical report, 1992. Preprint.

[Sch89] 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 Complements". *Artificial Intelligence*, 47, 1991.

[Vil85] M. B. Vilain. "The restricted language architecture of a hybrid representation system." In R. J. Bachmann, H. J. Levesque, R. Reiter (editors), *Proceedings of the 9th IJCAI*, pp. 547–551, Los Angeles, Cal., 1985.

5

An Empirical Analysis of Terminological Representation Systems

Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich[1]

ABSTRACT The family of terminological representation systems has its roots in the representation system KL-ONE. Since the development of this system more than a dozen similar representation systems have been developed by various research groups. These systems vary along a number of dimensions. In this paper, we present the results of an empirical analysis of six such systems. Surprisingly, the systems turned out to be quite diverse leading to problems when transporting knowledge bases from one system to another. Additionally, the runtime performance between different systems and knowledge bases varied more than we expected. Finally, our empirical runtime performance results give an idea of what runtime performance to expect from such representation systems. These findings complement previously reported analytical results about the computational complexity of reasoning in such systems.

5.1 Introduction

Terminological representation systems support the taxonomic representation of terminology for AI applications and provide reasoning services over the terminology. Such systems may be used as stand-alone information retrieval systems [Devanbu *et al.*, 1991] or as components of larger AI systems. Assuming that the application task is the configuration of computer systems [Owsnicki-Klewe, 1988], the terminology may contain *concepts* such as *local area network, workstation, disk-less workstation, file server*, etc. Further, these concepts are interrelated by specialization relationships and the specification of necessary and sufficient conditions. A *disk-less workstation* may be *defined* as a *workstation* that has no *disk* attached to it, for example. The main reasoning service provided by terminological representation systems is checking for inconsistencies in concept specifications and determining the specialization relation between concepts—the so-called *subsumption relation.*

The first knowledge representation system supporting this kind of representation and reasoning was KL-ONE [Brachman and Schmolze, 1985]. Meanwhile, the underlying framework has been adopted by various research groups, and more than a dozen terminological representation systems have been implemented [Patel-Schneider *et al.*, 1990]. These systems vary along a number of important dimensions, such as implementation status, expressiveness of the underlying representation language, completeness of the reasoning services, efficiency, user interface, interface functionality, and integration with other modes of reasoning.

Nowadays, it seems reasonable to build upon an existing terminological representation system instead of building one from scratch. Indeed, this was the idea in our project WIP, which is aimed at knowledge-based, multi-modal presentation of information such as oper-

[0]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.
An expanded version of this paper has been published as DFKI-Report RR-92-16 [Heinsohn *et al.*, 1992].
[1]e-mail: {heinsohn,kudenko,nebel,profi}@dfki.uni-sb.de

ating instructions [Wahlster *et al.*, 1991]. However, it was by no means clear which system to choose. For this reason, we analyzed a subset of the available systems empirically. It turned out that the effort we had to invest could have well been used to implement an additional prototypical terminological representation system. However, we believe that the experience gained is worthwhile, in particular concerning the implementation of future terminological representation systems and standard efforts in the area of terminological representation systems.

One of the main results of our study is that the differences in expressiveness between the existing systems are larger than one would expect considering the fact that all of them are designed using a common semantic framework. These differences led to severe problems when we transported knowledge bases between the systems. Another interesting result is the runtime performance data we obtained. These findings indicate (1) that the structure of the knowledge base can have a significant impact on the performance, (2) that the runtime grows faster than linearly in all systems, and (3) that implementations ignoring efficiency issues can be quite slow. Additionally, the performance data gives an idea of what performance to expect from existing terminological representation systems. These results complement the various analytical results on the computational complexity of terminological reasoning.

5.2 The Experiment

The empirical analysis can be roughly divided into two parts. [2] The first part covers qualitative facts concerning system features and expressiveness. In order to describe the latter aspect, we first developed a "common terminological language" that covers a superset of all terminological languages employed in the systems we considered. The analysis of the expressiveness shows that the intersection over all representation languages used in the systems is quite small.

In the second part we ran different test cases on the systems in order to check out the performance, completeness and the handling of problematical cases. We designed five different groups of experiments. The first group consists of tests dealing with cases that are not covered by the common semantic framework of terminological representation systems. The second group explores the degree of the inferential completeness of the systems for "easy" (i.e., polynomial) inferences. It should be noted that we did not try to design these tests in a systematic fashion by trying out all possible combinations of language constructs, though. The third group consists of problems which are known to be "hard" for existing systems. They give an impression of the runtime performance under worst-case conditions.

For the fourth group of experiments we used existing knowledge bases to get an idea of the runtime performance under "realistic" conditions. First, we manually converted the knowledge bases into the "common terminological language" mentioned above. Then, we implemented a number of translators that map the knowledge bases formulated using the "common terminological language" into system specific knowledge bases.

Although the results of the fourth group of experiments give some clues of what the behavior of the systems may be in applications, we had not enough data points to confirm some of the conjectures that resulted from this initial test under realistic conditions. Additionally, it was not evident in how far the translation, which is only approximate, influenced the performance. For this reason, a fifth group of experiments was designed. A number of

[2] The details of the experiment are given in a technical report [Heinsohn *et al.*, 1992].

knowledge bases were generated randomly with a structure similar to the structure of the realistic knowledge bases.

In general, we concentrated on the *terminological representation* part (also called *TBox*) of the systems. This means that we ignored other representation and reasoning facilities, such as facilities for maintaining and manipulating databases of objects (also called *ABox*) that are described by using the concepts represented in the terminological knowledge base. This concentration on the terminological component is partly justified by the fact that the terminological part is the one which participates in most reasoning activities of the entire system. Thus, run time performance and completeness of the terminological part can be generalized to the entire system—to a certain degree. However, the systems may (for efficiency reasons) use different algorithms for maintaining a database of objects, which may lead to a different behavior in this case. Nevertheless, even if the generalization is not valid in general, we get at least a feeling how the terminological parts perform.

As a final note, we want to emphasize that our empirical analysis was not intended to establish a ranking between the systems. For this purpose, it would be necessary to assign weights to the dimensions we compared, and this can only be done if the intended application has been fixed. Despite the fact that we analyzed only the terminological subsystems, the tests are not intended to be complete in any sense and there may be more dimensions that could be used to analyze the systems. Further, the results apply, of course, only to the system versions explicitly mentioned in the following section. The system developers of a number of systems have improved their systems since we made our experiment. So, the runtime performance may have changed.

5.3 Systems

There are a large number of systems which could have been included in an empirical analysis, e.g., KL-ONE, LILOG, NIKL, K-REP, KRS, KRYPTON, and YAK (see e.g. [Patel-Schneider *et al.*, 1990; Nebel *et al.*, 1991]). However, we concentrated on a relatively small number of systems. This does not mean that we feel that the sy stems we did not include (or mention) are not worthwhile to be analyzed. The only reason not to include all the systems was the limited amount of time available. We hope, however, that our investigation can serve as a starting point for future empirical analyses. The systems we picked for the experiment are: BACK [Peltason, 1991] (Version 4.2, pre-released), CLASSIC [Patel-Schneider *et al.*, 1991] (Version 1.02, released), KRIS [Baader and Hollunder, 1991] (Version 1.0, experimental), LOOM [MacGregor, 1991] (Version of May 1990, pre-released), MESON [Owsnicki-Klewe, 1988] (Version 2.0, released), and SB-ONE [Kobsa, 1991] (Version of January 1990, released).

The BACK system has been developed at the Technical University of Berlin by the KIT-BACK group as part of the Esprit project ADKMS. The main application is an information system about the financial and organizational structure of a company [Damiani *et al.*, 1990]. It is the only system among the ones we tested that is written in PROLOG. We tested the system on a Solbourne 601/32 using SICSTUS-PROLOG 2.1.

CLASSIC has been developed in the AI Principles Research Department at AT&T Bell Laboratories. It supports only a very limited terminological language, but turned out to be very useful for a number of applications [Devanbu *et al.*, 1991]. As all other systems except for BACK, it is written in COMMONLISP and we tested it on a MacIvory.

KRIS has been developed by the WINO project at DFKI. In contrast to other systems, it provides complete inference algorithms for very expressive languages. Efficiency considerations

have played no role in the development of the system.

LOOM has been developed at USC/ISI and supports a very powerful terminological logic—in an incomplete manner, though—and offers the user a very large number of features. In fact, LOOM can be considered as a programming environment.

MESON has been developed at the Philips Research Laboratories, Hamburg, as a KR tool for different applications, e.g., computer configuration [Owsnicki-Klewe, 1988]. Although it is also written in COMMONLISP, we tested it not on a MacIvory but on a Solbourne 601/32 in order to take advantage of its nice X-Window interface.

SB-ONE has been developed in the XTRA project at the University of Saarland as the knowledge representation tool for a natural language project. One of the main ideas behind the design of the system was the possibility of direct graphical manipulations of the represented knowledge.

5.4 Qualitative Results

The main qualitative result of our experiment is that although the systems were developed with a common framework in mind, they are much more diverse than one would expect. First of all, the terminological languages that are supported by the various systems are quite different. While three of the six systems use a similar syntactic scheme (similar to the one first used by Brachman and Levesque [Brachman and Levesque, 1984]), and one system adapted this syntactic scheme for PROLOG, i.e., infix instead of prefix notation, the remaining two systems use quite different syntactic schemes. Furthermore, there are not only superficial differences in the syntax, but the set of (underlying) *term-forming* operators varies, as well. In fact, the common intersection of all languages we considered is quite small. It contains only the concept-forming operators *concept conjunction*, *value restriction*, and *number restriction*.[3]

These differences led to severe problems when we designed automatic translators from the "common terminological language" to the languages supported by the different systems. Because of the differences in expressiveness, the translations could only be approximate, and because of the differences in the syntax we used a translation schema that preserved the meaning (as far as possible) but introduced a number of auxiliary concepts. Using the translated knowledge bases, we noticed that the introduction of auxiliary concepts influences the runtime performance significantly—a point we will return to.

Discounting the differences in syntax and expressiveness, one might expect that the common semantic framework (as spelled out by Brachman and Levesque [Brachman and Levesque, 1984]) leads to identical behavior on inputs that have identical meaning and match the expressiveness of the systems. However, this is unfortunately wrong. When a formal specification is turned into an implemented system, there are a number of areas that are not completely covered by the specification. One example is the order of the input. So, some systems allow for *forward references* in term definitions and some do not. Furthermore, some systems support *cyclic definitions* (without handling them correctly according to one of the possible semantics [Nebel, 1991], however, or permitting cyclic definitions only in some contexts), and some give an error message. Also *redefinitions* of terms are either marked as errors, processed as revisions of the terminology, or treated as incremental additions to the

[3] The technical report [Heinsohn *et al.*, 1992] contains tables specifying precisely the expressiveness of the different systems.

definition. Finally, there are different rules for *determining the syntactic category* of an input symbol.

Another area where designers of terminological systems seem to disagree is what should be considered as a an error by the user. So, some systems mark the definitions of *semantically equivalent* concepts as an error or refuse to accept *semantically empty* (inconsistent) concepts, for instance.

These differences between the systems made the translation from the "common terminological language" to system-specific languages even more complicated. In fact, some of the problems mentioned above were only discovered when we ran the systems on the translated knowledge bases. We solved that problem by putting the source form of the knowledge base into the most unproblematical form, if possible, or ignored problematical constructions (such as cyclic definitions) in the translation process.

Summarizing, these results show that the ongoing process of specifying a common language for terminological representation and reasoning systems [Neches *et al.*, 1991, p. 50–51] will probably improve the situation in so far as the translation of knowledge bases between different systems will become significantly easier. One main point to observe, however, is the area of pragmatics we touched above, such as permitting forward references.

Finally, we should mention a point which all systems had in common. In each system we discovered at least one deviation from the documentation, such as missing an obvious inference or giving a wrong error message. This is, of course, not surprising, but shows that standard test suites should be developed for these systems.

There are a number of other dimensions where the systems differ, such as the integration with other reasoning services, the functionality of graphical user interfaces, ease of installation, and user friendliness, but these are issues which are very difficult to evaluate.

5.5 Quantitative Results

One important feature of a representation and reasoning system is, of course, its runtime performance. In the case of terminological representation systems, the time to compute the *subsumption hierarchy* of concepts—a process that is often called *classification*—is an interesting parameter. In order to get a feeling for the runtime behavior of the systems we designed several tests to explore how the systems behave under different conditions. Since most of the systems are still under development, the runtime data we gathered is most probably not an accurate picture of the performance of the most recent versions of the systems. In particular, new (and faster) versions of BACK, CLASSIC, and LOOM are available.

Computational complexity results show that *subsumption determination* between *terms* is NP-hard [Donini *et al.*, 1991] or even undecidable [Schmidt-Schauß, 1989] for reasonably expressive languages. Even assuming that *term-subsumption* can be computed in polynomial time (e.g., for restricted languages), subsumption determination in a *terminology* is still NP-hard [Nebel, 1990]. In order to explore this issue, we designed some tests to determine the behavior of the systems under conditions that are known to be hard.

One test exploits the NP-hardness result for term-subsumption for languages that contain concept-conjunction, value restrictions, and qualified existential restrictions [Donini *et al.*, 1992]. It turned out that three systems could not express this case, one system reported an internal error, one system missed the inference (but exhibited a polynomial runtime behavior), and only one system handled the case, but with a very rapid growth in runtime.

Three other tests exploit the NP-hardness result for subsumption in terminologies [Nebel, 1990]. The first two tests show that only one of the six systems uses a naive way of performing subsumption in a terminology by expanding all concept definitions before checking subsumption [Nebel, 1990, p. 239]. The third test was designed in a way such that also clever subsumption algorithms are bound to use exponential time [Nebel, 1990, p. 245]. The results of the latter test are given in Figure 5.1.[4] They clearly indicate that the systems indeed exhibit a very rapid growth in the runtime.

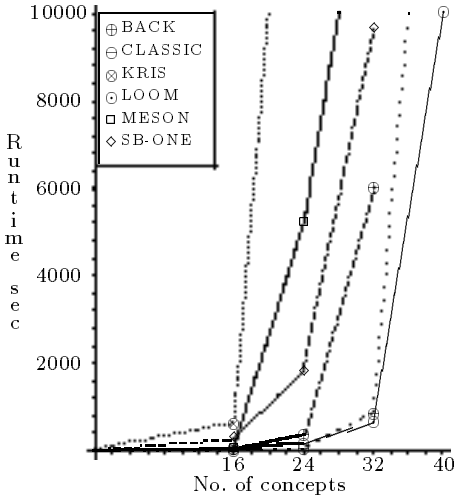

FIGURE 5.1. Runtime performance for hard cases

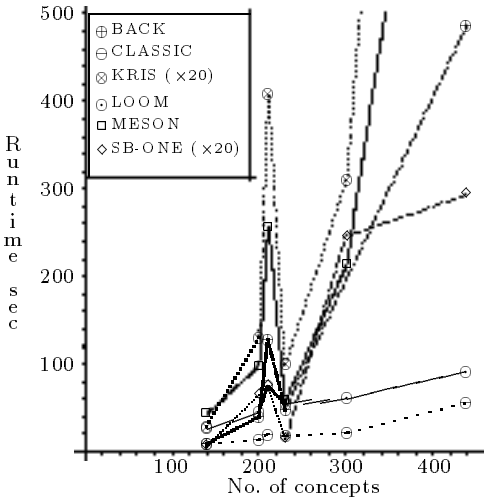

FIGURE 5.2. Runtime performance for realistic cases

Despite their theoretical intractability, terminological reasoning systems have been used for quite a while and the literature suggests that the knowledge bases involved were larger than just toy examples (i.e., more than 40 concepts). Hence, one would assume that the knowledge bases that have been used in applications are of a form that permits easy inferences, or the

[4] The runtimes of BACK and MESON are not directly comparable with the other systems because BACK and MESON were tested on a Solbourne 601/32, which is two to three times faster than a MacIvory with respect to the execution of CommonLisp programs, a remark that applies also to the other runtime performance tests. Additionally, it is not clear to us in how far the performance of BACK is influenced by the fact that it is implemented in PROLOG.

systems are incomplete and ignore costly inferences. In any case, it is questionable of whether the runtime performance for worst-case examples give us the right idea of how systems will behave in applications.

In order to get a feeling of the runtime performance under "realistic" conditions, we asked other research groups for terminological knowledge bases they use in their projects. Doing so, we obtained six different knowledge bases. As mentioned above, these were first manually translated into the "common terminological language" and then translated to each target language using our (semi-) automatic translators. In Figure 5.2, the runtime for the systems is plotted against the number of concepts defined in the different knowledge bases.

There are a number of interesting points to note here. First of all, two systems, namely, KRIS and SB-ONE, were too slow to be plotted together with the other systems using the same scale. For this reason, we divided the runtimes by the factor of 20 before plotting it.

Second, the diagram indicates that the runtime ratio between the slowest system (KRIS) and the fastest system (CLASSIC) in case of the largest knowledge base is extreme, namely, $45,000/56 \approx 800$. Considering that KRIS was developed as an experimental testbed for different complete subsumption algorithms and CLASSIC was designed as an efficient system for an expressively limited language to be used in different applications, this result is actually not completely surprising. It would be of course desirable to explain this and other differences in performance on the level of algorithms and implementation techniques. However, these issues are not described in the literature and a source code analysis was beyond the scope of our analysis.

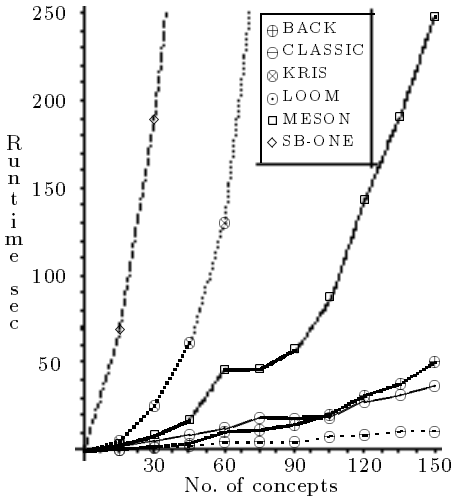

FIGURE 5.3. Runtime performance for small random KBs

Third, the knowledge base with 210 concepts seems to be somehow special because the runtime curve shows a peak at this point. Inspecting this knowledge base, we discovered that one concept is declared to be super-concept (i.e, mentioned literally in the definition) of 50% of all other concepts. Removing this concept led to a smoother curve. Hence, the structure of a knowledge base can severely influence the runtime. Although this should have been obvious already from the first diagram showing the runtime behavior under worst-case conditions, it is an indication that under realistic conditions the runtime behavior can be unexpectedly influenced by the structure of the knowledge base.

Summarizing the curves in Figure 5.2, it seems to be the case that most of the systems, except for SB-ONE, are similar in their runtime behavior in that the same knowledge bases

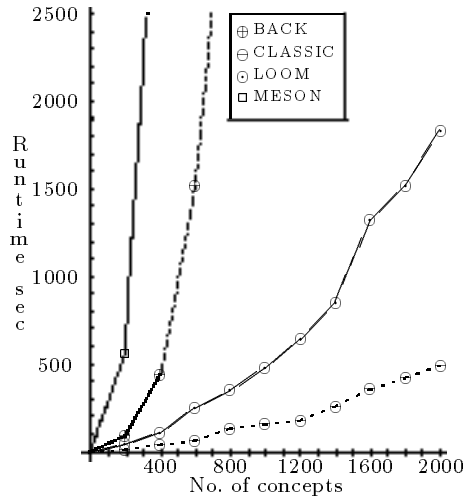

FIGURE 5.4. Runtime performance for large random KBs

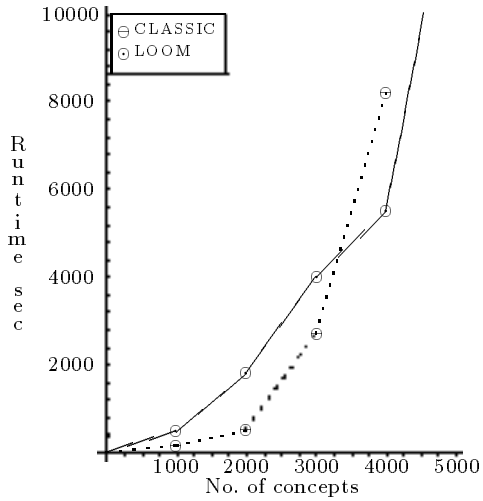

FIGURE 5.5. Runtime performance for very large random KBs

are considered as "difficult" or "easy" to a similar degree. However, it is not clear whether the system runtimes differ only by a constant factor or not. Further, because of the approximative nature of the translations and the introduction of auxiliary concepts, it is not clear to us how reliable the data is. For these reasons, we generated knowledge bases randomly in the intersection of all languages—avoiding the translation problem. The structure of these generated knowledge bases resembles the structure of the six real knowledge bases (percentage of defined concepts, average number of declared super-concepts, average number of role restrictions, etc.). The results of this test are given in Figures 5.3, 5.4, and 5.5.

Comparing the curves in these three figures with the curves in Figure 5.2, it seems to be the case that the structure of the randomly generated knowledge bases is indeed similar to the structure of realistic knowledge bases in so far as they lead to a similar runtime performance. However, we do not claim that the knowledge bases are realistic with respect to all possible aspects. In fact, too few facts are known about which structural properties can influence the performance of terminological representation systems. Bob MacGregor, for instance, reported that the number of distinct roles heavily influence the performance. He observed that the runtime *decreases* when the number of distinct roles is increased and all

other parameters are hold constant (same number of concepts and role restrictions).

These curves indicate that the runtime grows faster than linearly with the number of concepts. We conjecture that in general the runtime of terminological representation systems is at least quadratic in the number of concepts. This conjecture is reasonable because identifying a partial order over a set of elements that are ordered by an underlying partial order is worst-case quadratic (if all elements are incomparable), and there is no algorithm known that is better for average cases. In fact, average case results are probably very hard to obtain because it is not known how many partial orders exist for a given number of elements [Aigner, 1988, p. 271].

From this, we conclude that designing efficient terminological representation systems is not only a matter of designing efficient *subsumption algorithms*, but also a matter of designing efficient *classification algorithms*, i.e., fast algorithms that construct a partial order. The main point in this context is to minimize the number of subsumption tests.

Another conclusion of our runtime tests could be that the more expressive and complete a system is, the slower it is—with KRIS as a system supporting complete inferences for a very expressive language and CLASSIC with almost complete inferences for a comparably simple language at the extreme points. However, we do not believe that this is a *necessary* phenomenon. A desirable behavior of such systems is that the user would have "to pay only as s/he goes," i.e., only if the full expressive power is used, the system is slow. In fact, at DFKI together with the WINO group we are currently working on identifying the performance bottlenecks in the KRIS system. First experiences indicate that it is possible to come close to the performance of LOOM and CLASSIC for the knowledge bases used in our tests.

5.6 Conclusions

We have analyzed six different terminological representation and reasoning systems from a qualitative and quantitative point of view. The empirical analysis of the different terminological languages revealed that the common intersection of the languages supported by the systems is quite small. Together with the fact that the systems behave differently in areas that are not covered by the common semantic framework, sharing of knowledge bases between the systems does not seem to be easily achievable. In fact, when we tried to translate six different knowledge bases from a "common terminological language" into the system-specific languages we encountered a number of problems.

Testing the runtime performance of the systems, we noted that the structure of the knowledge base can have a significant impact on the performance, even if we do not consider artificial worst-case examples but real knowledge bases. Further, the systems varied considerably in their runtime performance. For instance, the slowest system was approximately 1000 times slower than the fastest in one case. The overall picture suggests that for all systems the runtime grows at least quadratically with the size of the knowledge base. These findings complement the various analyses of the computational complexity, providing a user of terminological systems with a feeling of how much he can expect from such a system in reasonable time.

5.7 Acknowledgments

We would like to thank the members of the research groups KIT-BACK at TU Berlin, AI Principles Research Department at Bell Labs., WINO at DFKI, LOOM at USC/ISI, MESON at Philips Research Laboratories, and XTRA at the University of Saarland, for making their systems and/or knowledge bases available to us, answering questions about their systems, and providing comments on an earlier version of this paper. Additionally we want to express our thanks to Michael Gerlach, who made one of the knowledge bases available to us.

5.8 References

[Aigner, 1988] Aigner, Martin 1988. *Combinatorical Search*. Teubner, Stuttgart, Germany.

[Baader and Hollunder, 1991] Baader, Franz and Hollunder, Bernhard 1991. KRIS: Knowledge representation and inference system. *SIGART Bulletin* 2(3):8–14.

[Brachman and Levesque, 1984] Brachman, Ronald J. and Levesque, Hector J. 1984. The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference of the American Association for Artificial Intelligence*, Austin, TX. 34–37.

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

[Damiani *et al.*, 1990] Damiani, M.; Bottarelli, S.; Migliorati, M.; and Peltason, C. 1990. Terminological Information Management in ADKMS. In *ESPRIT '90 Conference Proceedings*, Dordrecht, Holland. Kluwer.

[Devanbu *et al.*, 1991] Devanbu, Premkumar T.; Brachman, Ronald J.; Selfridge, Peter G.; and Ballard, Bruce W. 1991. LaSSIE: a knowledge-based software information system. *Communications of the ACM* 34(5):35–49.

[Donini *et al.*, 1991] Donini, Francesco M.; Lenzerini, Maurizio; Nardi, Daniele; and Nutt, Werner 1991. The complexity of concept languages. In Allen, James A.; Fikes, Richard; and Sandewall, Erik, editors 1991, *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, Cambridge, MA. Morgan Kaufmann. 151–162.

[Donini *et al.*, 1992] Donini, Francesco M.; Lenzerini, Maurizio; Nardi, Daniele; Hollunder, Bernhard; Nutt, Werner; and Spacamella, Alberto Marchetti 1992. The complexity of existential quantification in concept languages. *Artificial Intelligence* 53(2-3):309–327.

[Heinsohn *et al.*, 1992] Heinsohn, Jochen; Kudenko, Daniel; Nebel, Bernhard; and Profitlich, Hans-Jürgen 1992. An empirical analysis of terminological representation systems. DFKI Research Report RR-92-16, German Research Center for Artificial Intelligence (DFKI), Saarbrücken.

[Kobsa, 1991] Kobsa, Alfred 1991. First experiences with the SB-ONE knowledge representation workbench in natural-language applications. *SIGART Bulletin* 2(3):70–76.

[MacGregor, 1991] MacGregor, Robert 1991. Inside the LOOM description classifier. *SIGART Bulletin* 2(3):88–92.

[Nebel *et al.*, 1991] Nebel, Bernhard; von Luck, Kai; and Peltason, Christof, editors 1991. International workshop on terminological logics. DFKI Document D-91-13, German Research Center for Artificial Intelligence (DFKI), Saarbrücken. Also published as KIT Report, TU Berlin, and IWBS Report, IBM Germany, Stuttgart.

[Nebel, 1990] Nebel, Bernhard 1990. Terminological reasoning is inherently intractable. *Artificial Intelligence* 43:235–249.

[Nebel, 1991] Nebel, Bernhard 1991. Terminological cycles: Semantics and computational properties. In Sowa, John F., editor 1991, *Principles of Semantic Networks*. Morgan Kaufmann, San Mateo, CA. 331–362.

[Neches *et al.*, 1991] Neches, Robert; Fikes, Richard; Finin, Tim; Gruber, Thomas; Patil, Ramesh; Senator, Ted; and Swartout, William R. 1991. Enabling technology for knowledge sharing. *The AI Magazine* 12(3):36–56.

[Owsnicki-Klewe, 1988] Owsnicki-Klewe, Bernd 1988. Configuration as a consistency maintenance task. In Hoeppner, Wolfgang, editor 1988, *Künstliche Intelligenz. GWAI-88, 12. Jahrestagung*, Eringerfeld, Germany. Springer-Verlag. 77–87.

[Patel-Schneider *et al.*, 1990] Patel-Schneider, Peter F.; Owsnicki-Klewe, Bernd; Kobsa, Alfred; Guarino, Nocola; MacGregor, Robert; Mark, William S.; McGuinness, Deborah; Nebel, Bernhard; Schmiedel, Albrecht; and Yen, John 1990. Term subsumption languages in knowledge representation. *The AI Magazine* 11(2):16–23.

[Patel-Schneider *et al.*, 1991] Patel-Schneider, Peter F.; McGuinness, Deborah L.; Brachman, Ronald J.; Alperin Resnick, Lori; and Borgida, Alex 1991. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin* 2(3):108–113.

[Peltason, 1991] Peltason, Christof 1991. The BACK system – an overview. *SIGART Bulletin* 2(3):114–119.

[Schmidt-Schauß, 1989] Schmidt-Schauß, Manfred 1989. Subsumption in KL-ONE is undecidable. In Brachman, Ron J.; Levesque, Hector J.; and Reiter, Ray, editors 1989, *Principles of Knowledge Representation and Reasoning: Proceedings of the 1st International Conference*, Toronto, ON. Morgan Kaufmann. 421–431.

[Wahlster *et al.*, 1991] Wahlster, Wolfgang; Andre, Elisabeth; Bandyopadhyay, Som; Graf, Winfried; and Rist, Thomas 1991. WIP: the coordinated generation of multimodal presentations from a common representation. In Ortony, Andrew; Slack, John; and Stock, Oliviero, editors 1991, *Computational Theories of Communication and their Applications*. Springer-Verlag, Berlin, Heidelberg, New York. To appear. Also available as DFKI Research Report RR-91-08.

6

Knowledge Representation for Natural Language Processing Agents

Elizabeth A. Hinkelman[1]

ABSTRACT The DISCO project makes use of extensive inference and planning facilities to support work in natural language pragmatics, including a model of the appointment scheduling application domain. This paper gives an overview of knowledge representation techniques used. It gives one formulation of two central ideas, shared beliefs and n-way reliable speech acts. It concludes with a summary of knowledge representation system selection criteria and open issues in knowledge representation.

6.1 Introduction

The project on DIalogue Systems for COoperating agents, DISCO, conducts research on natural language processing for systems that function autonomously in an environment containing several other agents. It aims at modelling conversations such as the following:

- A: Remember our appointment with Siemens?
- B: yes
- C: yes
- A: They want to postpone it.
- C: How about next Thursday?
- B: I'll be in Berlin next week.

...

- C: Good. I'll suggest that to Siemens.

An autonomous cooperative software system with natural language capabilities might take the role of C in this conversation. Acting as an appointment secretary, it would keep track of appointments and contact participants to make arrangements. Full deployment of natural language capabilities in this domain requires knowledge representation for

- equality reasoning, for NL referring expressions
- temporal reasoning, for NL tense and calendrical calculations
- planning, for NL dialogue and the appointment domain

[1] Project DISCO, DFKI Saarbruecken, email: hinkelma@dfki.uni-sb.de. This work was supported by the Ministry for Research and Technology, Federal Republic of Germany, under grant number ITW 9002 0 to DFKI inc.

- plan recognition, for NL dialogue and appointment domain
- modal operators for belief and intent

Central to the DISCO approach is the concept that communication can be modelled as actions just like other actions in the domain, and can be processed with the same basic techniques[3]. A full model of such actions must be cast in terms of dialogue participants' beliefs, since although the goal of an utterance may be to affect an agent's behavior, this is always mediated by belief. Note that in any given dialogue, the number and ordering of communicative actions cannot be known in advance; rather, participants must react to the situation as it develops. Feedback is nearly immediate, and plans must be recast frequently. Plan recognition must be capable of processing plans that are not fully specified in advance.

The general architecture used by DISCO for such applications is shown in Figure 6.1. The bulk of linguistic analysis and generation is performed within a typed feature structure formalism. Only dialogue facilities (and domain modelling) make use of general inference capacity.

An incoming utterance is parsed, assigned a compositional semantics, and analyzed into a set of speech act descriptions. These are translated from feature structures into STRIPS-style operators with explicit, propositional preconditions and effects. The figure shows a set of three alternative speech act interpretations, resembling "bonbons", being disambiguated within the current dialogue context. The preferred interpretation, which for our sample dialogue's first utterance would be a yes/no question, becomes the input for plan recognition and planning processes. The system obtains an answer to the question, which involves issuing a query to an external database for appointments with Siemens. As it happens, a unique answer is found and the system initiates generation of a positive response.

Note that the interface to the external calendar database is mediated by an internal model of its functionality. While databases may be implemented completely internally to the dialogue system, as is the email address locator illustrated in the figure, the general technique is exercised here because it is required for interfacing with other external applications such as servomotors and sensors.

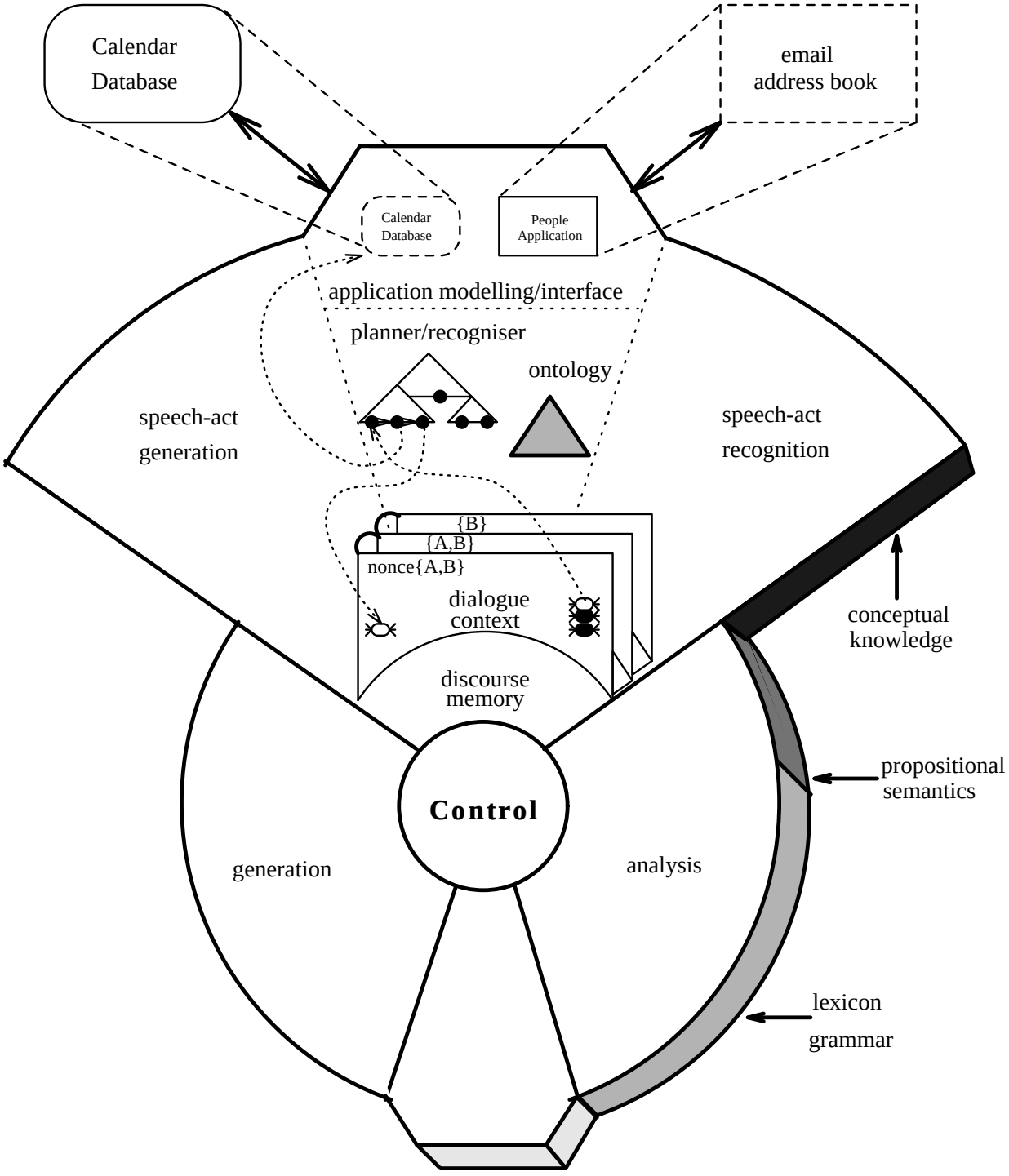

FIGURE 6.1. Calendar Application, Sept. 1993 (Planned)

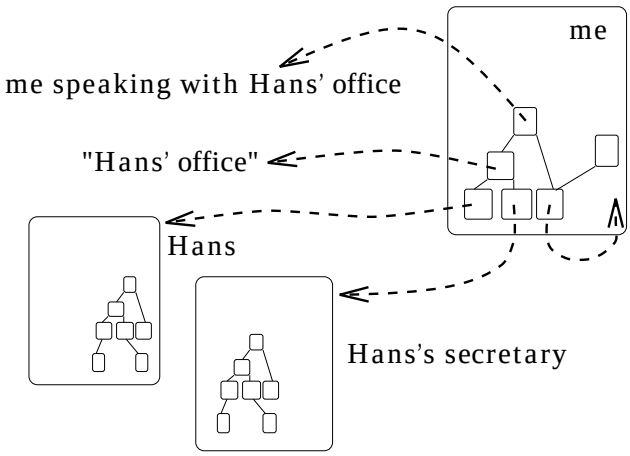

FIGURE 6.2. Some Corporate Agents

6.2 Multiagent Attitude Attribution

An individual agent which participates in multiagent dialogues must have the ability to model the dialogue situation, including beliefs and goals of other agents, and shared beliefs and goals. The approach we use to attitude attribution was developed in [8] and has as its primary characteristics

- attitudes are attributed *directly* to groups. That is, an individual agent models the beliefs of a group not as the intersection of the beliefs of the group members, but as a separate set of beliefs which may, under anomalous circumstances, be held by none of the members.
- attitudes of groups are modelled as propagating by default to and from the beliefs of members.
- default propagation of attitudes is constrained by semantic relevance conditions specific to the attitude.
- average case complexity is thereby reduced, and fixed-point constructions are avoided.

This is illustrated in Figure 6.2. I as an individual agent model the beliefs and goals of the individuals Hans, Hans' secretary, and myself. Note that my model of my own attitudes may differ from my actual working attitudes. I also model two collective or "corporate" agents: "Hans's office", consisting of Hans and the secretary, and a temporary agent which is my conversation with Hans's office. These latter are social constructs, which unlike individuals don't actually have attitudes, though individual agents model them as doing so.

The attitude propagation rules are given more precisely in Figure 6.3.

Assuming	
s	is an arbitrary agent
c	is a corporate agent containing s
active c	agent c is functioning normally
role c s = r	agent s has the role r in c

Downward case	
Ascribe attitude a to agent s if consistent and:	
$_{c}$ a	agent c holds a
drel r a	attitude a is 'downward relevant' to r

Upward case	
Ascribe attitude b to agent c is consistent and:	
$_{s}$ b	agent s holds b
urel r b	attitiude b is 'upward relevant' to r
s ≠ **me**	the attitude does not come from me (this forces explicit communication)

FIGURE 6.3. Attitude Propagation Rules

6.3 N-Way, Reliable Speech Acts

Having endowed agents with the ability to attribute attitudes to groups, it is now possible to provide them with the means for reasoning about (and performing) group communication. An agent should be able to plan activities in some arbitrary task domain, which happens to involve some communication, but without having to plan the exact dialogue structure. To this end, the communication model of n-way, reliable speech acts [3] has been developed. Its principal characteristics are

- speech acts are represented as STRIPS-style operators, with preconditions and effects expressed in propositional form.
- the agent of the action is a single corporate agent, and the action so represented is therefore a true joint action.
- preconditions and effects are expressed in terms of this corporate agent's attitudes.
- speech acts are not complete until they are acknowledged.
- member agents are committed to a specific process for resolving communication failures[9].

The abstraction of reliable group communication results.

A reliable speech act is now decomposable rather than being an atomic action. It consists of an *initiation*, an *acknowledgement*, and as many other specialized subcomponents as are necessary to resolve interpretation problems. The resolution process was previously modelled with finite-state automata ([9]); we propose to treat it as a productive planning process.

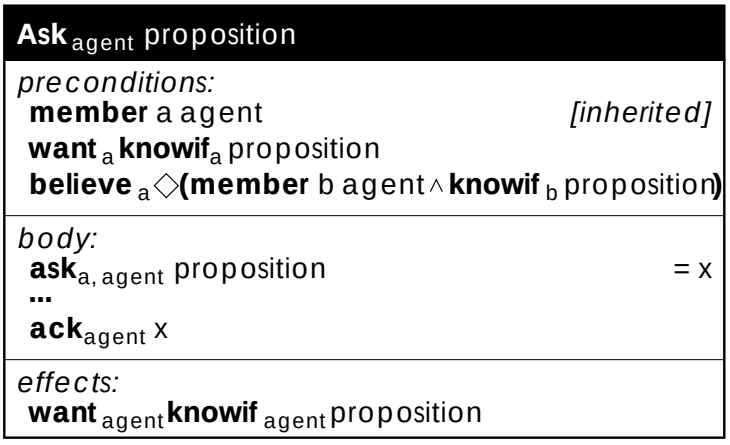

FIGURE 6.4. An N-Way, Reliable Ask Act

Figure 6.4 shows a reliable yes/no question, whose group agent is *agent* and whose initiating agent is *a*. The body consists of an initiating action, namely *a* uttering the question, and ultimately group acknowledgement of this utterance.

Intuitively, the preconditions for this group action are that *a* is a member of *agent*, wants to know (for lack of a better word) whether some proposition is true, and believes it possible that some other member *b* of *agent* knows whether the proposition holds. The effect on participants is that they will model the group as wanting to know whether the proposition holds. Should a member have an answer, the member will subsequently infer a goal to give it.

Thus, we must allow an agent who does not have a belief about the truth value of a proposition to represent that another agent does. For truth values, this is the relatively straightforward *knowif* predicate defined in the figure. It expresses correspondence between a state of affairs and the second agent's beliefs, without selecting the state. This predicate is only used within some agent's beliefs; thus the state of affairs is a subjective reality.

Both this construct and the much bigger technical hack needed for wh-questions can be found in [1]. In brief, wh-questions require *knowref*, which relies on a particular interpretation of quantification into the belief operator in order to force the agent to know a referential identity of a variable in a referring expression. The referent is also required to be unique.

The action initiating a reliable speech act serves two purposes. One is to bear the type and content of the speech act. The other is its immediate effect, the introduction of a group goal that will trigger an acknowledgement. In Figure 6.5, the *Understand* predicate holds when an individual is able to construct a well-formed interpretation for the utterance in context.

The course of inference for the hearer of an initial ask (containing an appointment description or time new to this conversation) is shown in Figure 6.6.

ask$_{a,\ agent}$ proposition	= x
preconditions: **<communications medium ok>** **believe** $_a$ proposition	*[inherited]*
effects: **want**$_{agent}$ **knowif**$_{agent}$ **understand** $_{agent}$ x	*[inherited]*

FIGURE 6.5. Initial ask Action

1) infer own feedback goal from ask's group goal
2) infer own information goal from Ask's group goal
3) plan to answer question; do I know the answer?
   - 4y) yes; execute inform of answer (=feedback)
   - 4n) no; can I plan to find out?
     - 5n) no; inform that I don't know (=feedback)
     - 5y) yes; acknowledge if this will be slow
       6) execute calendar database query
       7) execute inform of result (=feedback)

FIGURE 6.6. Inference from Initial ask

6.4 Issues for KR Systems

In addition to the knowledge representation tasks discussed above, the DISCO project sees three classes of issues for design and selection of knowledge representation systems.

6.4.1 Practical Issues

In selecting a knowledge representation system for implementation of natural language processing agents, certain practical concerns cannot be avoided. One tedious but important concern for the DISCO project was timeframe. The fourteen months remaining at the system selection point spoke in favor of a relatively mature product, as a prerequisite for experimentation and development work. Equally important was the availability of fulltime staff support, with brisk turnaround on bug reports.

A more substantive issue was system support for agentive action, encompassing both a logic of action and some means of performing actions. In the end, this was achieved by selecting a knowledge representation system with a restricted logic of action, a full logic programming language, and a full interface with underlying Common Lisp (CL includes interprocess communication facilities). Access to complete programming languages also made possible the many pieces of customization necessary for application work.

Processing efficiency has not been an issue to date, but may become one. Fundamentally, LISP-based implementations are not efficient.

+ tempos, timelogic [Allen, Koomen]
+ plan recognition [Kautz]
+ equality reasoning
+ structured types and inheritance
+ truth maintenance
+ belief spaces
+ horn clause theorem prover
+ lisp interfaces

- several of the above needed extensions
- no planner
- - worse syntax than Common Lisp
- - fairly buggy

FIGURE 6.7. The Rhetorical KR System

6.4.2 Functional Issues

The knowledge representation system chosen to support dialogue work in DISCO was designed specifically for the purpose [6]. Figure 6.7 summarizes its advantages and disadvantages.

The temporal logic was augmented locally to handle calendrical calculations and their association with the actual current time. Multiagent belief attribution was built on system-supplied reasoning contexts. Multiagent speech acts were built using these and plan representation facilities.

Simple STRIPS-style planning is being built locally with the intention of replanning after every utterance. Some allowance may be made for re-use of previous results. The state of the art in reactive planning can be summarized as follows:

- robotics: is paying increasing attention to notions of feedback and integration of plan execution and planning; however, such work is oriented specifically toward motion planning and physical feedback, as [2].
- "situated AI": researchers in this area originally claimed that action did not require logical representation or planning at all but was purely a collection of stimulus-response pairs. They have been backing off rapidly since; see [4; 5].
- dialogue systems: tend not to work for more than two examples, which can then be based on prespecified standard scenarios. The most interesting recent variation, driven by the move to speech systems based on markov models, represents the standard scenario statistically [7].

6.4.3 Longterm Research Issues

In the two to four year timeframe, DISCO and its successors will struggle with several larger issues for KR formalisms.

- Integration of knowledge representation facilities with the typed feature structure formalism used to represent detailed linguistic structures. Currently this is done with expensive and incomplete translation processes, at both the formalism and content levels. We are also experimenting with a regimen of backtracking across both feature-structure based and logic-based modules. However, it should be possible to design formalisms for linguistic and general AI that can be related to each other in a much cleaner and deeper way. Constraint logic programming is of interest here.

- Preference, search, and indexing. Declarative formalisms are typically intended for use in representing some set of equally valid solutions; in processing, however, some are identified more quickly or are otherwise selected over others. Prolog-style reliance on ordering is inferior to explicit treatment of preferences.

- Softening failures. Perhaps most importantly, symbolic formalisms like logic programs and feature structures lead to brittle applications, that either find a solution or fail. Notions of preference can in principle be used to order solution candidates even when they are nonoptimal or not completely correct. For instance, in parsing one might like to ignore failures of agreement, when no other interpretation is available. In dialogue, one would like the introduction of a new sort of object, or a false presupposition, or an inappropriate speech act, not to lead to system crash.

More specific areas of longterm interest include belief revision and models of evidence (since the purpose of communication is often to change rather than augment the beliefs of the hearer).

6.5 References

[1] James Allen. Recognizing intentions from natural language utterances. In Michael Brady and Robert C. Berwick, editors, *Computational Models of Discourse*. MIT Press, 1983.

[2] James R. Firby. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University, 1989.

[3] Elizabeth A. Hinkelman and Stephen P. Spackman. Abductive Speech Act Recognition, Corporate Agents and the COSMA System. In W. J. Black, G. Sabah, and T. J. Wachtel, editors, *Abduction, Beliefs and Context: Proceedings of the second ESPRIT PLUS workshop in computational pragmatics*, 1992.

[4] Leslie Pack Kaelbling. Compiling operator descriptions into reactive strategies using goal regression. Technical report, Teleos Research, Palo Alto, CA USA, 1990.

[5] Pattie Maes. Situated agents can have goals. *Robotics and Autonomous Systems*, 6(1,2):49–70, 1990.

[6] Bradford W. Miller. The rhetorical knowledge representation system reference manual. Technical Report TR326, Computer Science Dept. University of Rochester, November 1990.

[7] Masaaki Nagata. Using pragmatics to rule out recognition errors in cooperative task-oriented dialogues. In *Proc International Conference on Spoken Language Processing*, pages 647–650, Canada, 1992. University of Alberta.

[8] Stephen P. Spackman and Elizabeth A. Hinkelman. Corporate Agents. 1993. DFKI technical report.

[9] David R. Traum and Elizabeth A. Hinkelman. Conversation Acts in Task-oriented Spoken Dialogue. *Computational Intelligence*, 8(3):575–599, 1992. Special Issue on Non-literal language.

7

Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra (Short Version)

Bernhard Nebel[1]
Hans-Jürgen Bürckert[2]

ABSTRACT We introduce a new subclass of Allen's interval algebra we call "ORD-Horn subclass," which is a strict superset of the "pointisable subclass." We prove that reasoning in the ORD-Horn subclass is a polynomial-time problem and show that the path-consistency method is sufficient for deciding satisfiability. Further, using an extensive machine-generated case analysis, we show that the ORD-Horn subclass is a maximal tractable subclass of the full algebra (assuming P≠NP). In fact, it is the unique greatest tractable subclass amongst the subclasses that contain all basic relations.

7.1 Introduction

Temporal information is often conveyed qualitatively by specifying the relative positions of time intervals such as "...point to the figure while explaining the performance of the system ..." Further, for natural language understanding [3; 18], general planning [4; 6], presentation planning in a multi-media context [7], and knowledge representation [25], the representation of qualitative temporal relations and reasoning about them is essential. Allen [2] introduces an algebra of binary relations on intervals for representing qualitative temporal information and addresses the problem of reasoning about such information. Since the reasoning problems are NP-hard for the full algebra [23], it is very unlikely that other polynomial-time algorithms will be found that solve this problem in general. Subsequent research has concentrated on designing more efficient reasoning algorithms, on identifying tractable special cases, and on isolating sources of computational complexity [8; 11; 17; 19; 20; 21; 22; 23; 24].

We extend these previous results in three ways. Firstly, we present a new tractable subclass of Allen's interval algebra, which we call *ORD-Horn subclass*. This subclass is considerably larger than all other known tractable subclasses (it contains 10% of the full algebra) and strictly contains the *pointisable subclass* [11; 20]. Secondly, we show that the *path-consistency method* is sufficient for deciding satisfiability in this subclass. Thirdly, using an extensive machine-generated case analysis, we show that this subclass is a maximal subclass such that satisfiability is tractable (assuming P≠NP).

From a practical point of view, these results imply that the path-consistency method has a much larger range of applicability than previously believed, provided we are mainly interested in satisfiability. Further, our results can be used to design backtracking algorithms for the full algebra that are more efficient than those based on other tractable subclasses.

[1] Project WIP, DFKI Saarbrücken, email: nebel@dfki.uni-sb.de
[2] Project TACOS, DFKI Saarbrücken, email: hjb@dfki.uni-sb.de

7.2 Reasoning about Interval Relations using Allen's Interval Algebra

Allen's [2] approach to reasoning about time is based on the notion of *time intervals* and *binary relations* on them. A **time interval** X is an ordered pair (X^-, X^+) such that $X^- < X^+$, where X^- and X^+ are interpreted as points on the real line.[3] So, if we talk about **interval interpretations** or I**-interpretations** in the following, we mean mappings of time intervals to pairs of distinct real numbers such that the beginning of an interval is strictly before the ending of the interval.

Given two interpreted time intervals, their relative positions can be described by *exactly one* of the elements of the set $\mathbf{B}$ of thirteen **basic interval relations** (denoted by B in the following), where each basic relation can be defined in terms of its **endpoint relations** (see Table 7.1). An atomic formula of the form XBY, where X and Y are intervals and $B \in \mathbf{B}$, is said to be **satisfied** by an I-interpretation iff the interpretation of the intervals satisfies the endpoint relations specified in Table 7.1.

Basic Interval Relation	Sym-bol	Pictorial Example	Endpoint Relations
X before Y	$\prec$	xxx yyy	$X^- < Y^-$, $X^- < Y^+$, $X^+ < Y^-$, $X^+ < Y^+$
Y after X	$\succ$		
X meets Y	m	xxxx yyyy	$X^- < Y^-$, $X^- < Y^+$, $X^+ = Y^-$, $X^+ < Y^+$
Y met-by X	$\mathsf{m}^\smile$		
X overlaps Y	o	xxxx yyyy	$X^- < Y^-$, $X^- < Y^+$, $X^+ > Y^-$, $X^+ < Y^+$
Y overlapped-by X	$\mathsf{o}^\smile$		
X during Y	d	xxx yyyyyyy	$X^- > Y^-$, $X^- < Y^+$, $X^+ > Y^-$, $X^+ < Y^+$
Y includes X	$\mathsf{d}^\smile$		
X starts Y	s	xxx yyyyyyy	$X^- = Y^-$, $X^- < Y^+$, $X^+ > Y^-$, $X^+ < Y^+$
Y started-by X	$\mathsf{s}^\smile$		
X finishes Y	f	xxx yyyyyyy	$X^- > Y^-$, $X^- < Y^+$, $X^+ > Y^-$, $X^+ = Y^+$
Y finished-by X	$\mathsf{f}^\smile$		
X equals Y	$\equiv$	xxxx yyyy	$X^- = Y^-$, $X^- < Y^+$, $X^+ > Y^-$, $X^+ = Y^+$

TABLE 7.1. The set $\mathbf{B}$ of the thirteen basic relations. The endpoint relations $X^- < X^+$ and $Y^- < Y^+$ that are valid for all relations have been omitted.

In order to express indefinite information, unions of the basic interval relations are used, which are written as sets of basic relations leading to 2^{13} binary **interval relations** (denoted by R, S, T)—including the **null relation** $\emptyset$ (also denoted by $\bot$) and the **universal relation** $\mathbf{B}$ (also denoted by $\top$). The set of all binary interval relations $2^{\mathbf{B}}$ is denoted by $\mathcal{A}$.

An atomic formula of the form $X\{B_1, \ldots, B_n\}Y$ (denoted by ϕ) is called **interval formula**. Such a formula is satisfied by an I-interpretation $\Im$ iff $X\ B_i\ Y$ is satisfied by $\Im$ for some i, $1 \leq i \leq n$. Finite sets of interval formulas are denoted by Θ. Such a set Θ is called I**-satisfiable** iff there exists an I-interpretation $\Im$ that satisfies every formula of Θ. Further, such a satisfying I-interpretation $\Im$ is called I**-model** of Θ. If an interval formula ϕ is

[3] Other underlying models of the time line are also possible, e.g., the rationals [5; 10]. For our purposes these distinctions are not significant, however.

satisfied by every I-model of a set of interval formulas Θ, we say that ϕ is **logically implied** by Θ, written $\Theta \models_I \phi$.

Fundamental **reasoning problems** in this framework include [8; 11; 21; 23]: Given a set of interval formulas Θ,

1. decide whether there exists an I-model of Θ (ISAT),
2. determine for each pair of intervals X, Y the *strongest implied relation* between them (ISI), i.e., the smallest set R such that $\Theta \models_I XRY$.

In the following, we often consider **restricted reasoning problems** where the relations used in interval formulas in Θ are only from a subclass $\mathcal{S}$ of all interval relations. In this case we say that Θ **is a set of formulas over** $\mathcal{S}$, and we use a parameter in the problem description to denote the subclass considered, e.g., ISAT($\mathcal{S}$). As is well-known, ISAT and ISI are equivalent with respect to polynomial Turing-reductions [23] and this equivalence also extends to the restricted problems ISAT($\mathcal{S}$) and ISI($\mathcal{S}$), provided $\mathcal{S}$ contains all basic relations.

Proposition 1 ISAT($\mathcal{S}$) *and* ISI($\mathcal{S}$) *are equivalent under polynomial Turing-reductions, provided* $\mathcal{S}$ *contains all basic relations.*[4]

The most prominent method to solve these problems (approximately for all interval relations or exactly for subclasses) is *constraint propagation* [2; 11; 17; 20; 22; 23] using a slightly simplified form of the *path-consistency algorithm* [13; 15]. In the following, we briefly characterize this method without going into details, though. In order to do so, we first have to introduce Allen's interval algebra.

Allen's interval algebra [2] consists of the set $\mathcal{A} = 2^{\mathbf{B}}$ of all binary interval relations and the operations unary **converse** (denoted by $\cdot^{\smile}$), binary **intersection** (denoted by $\cap$), and binary **composition** (denoted by $\circ$), which are defined as follows:[5]

$$\begin{array}{lrcl} \forall X,Y: & X R^{\smile} Y & \leftrightarrow & YRX \\ \forall X,Y: & X\,(R \cap S)\,Y & \leftrightarrow & XRY \wedge XSY \\ \forall X,Y: & X\,(R \circ S)\,Y & \leftrightarrow & \exists Z\!:\!(XRZ \wedge ZSY). \end{array}$$

Assume an operator Γ that maps finite sets of interval formulas to finite sets of interval formulas in the following way:

$$\begin{array}{rcl} \Gamma(\Theta) & = & \Theta \cup \\ & & \{X \top Y \mid X, Y \text{ appear in } \Theta\} \cup \\ & & \{XRY \mid (Y\, R^{\smile}\, X) \in \Theta\} \cup \\ & & \{X\,(R \cap S)\,Y \mid (XRY), (XSY) \in \Theta\} \cup \\ & & \{X\,(R \circ S)\,Y \mid (XRZ), (ZSY) \in \Theta\}. \end{array}$$

Since there are only finitely many different interval formulas for a finite set of intervals and Γ is monotone, it follows that for each Θ there exists a natural number n such that $\Gamma^n(\Theta) = \Gamma^{n+1}(\Theta)$. $\Gamma^n(\Theta)$ is called the **closure** of Θ, written $\overline{\Theta}$. Considering the formulas of the form $(XR_iY) \in \overline{\Theta}$ for given X, Y, it is evident that the R_i's are closed under intersection,

[4] Proofs are given in the long paper [16].

[5] Note that we obtain a relation algebra if we add *complement* and *union* as operations [11]. For our purposes, this is irrelevant, however.

and hence there exists $(XSY) \in \overline{\Theta}$ such that S is the *strongest relation* amongst the R_i's, i.e., $S \subseteq R_i$, for every i. The subset of a closure $\overline{\Theta}$ containing for each pair of intervals only the strongest relations is called the **reduced closure** of Θ and is denoted by $\widehat{\Theta}$.

As can be easily shown, every reduced closure of a set Θ is **path consistent** [13], which means that for every three intervals X, Y, Z and for every interpretation $\Im$ that satisfies $(XRY) \in \widehat{\Theta}$, there exists an interpretation $\Im'$ that agrees with $\Im$ on X and Y and in addition satisfies $(XSZ), (ZS'Y) \in \widehat{\Theta}$. Under the assumption that $(XRY) \in \Theta$ implies $(Y\ R^{\smile}\ X) \in \Theta$, it is also easy to show that path consistency of Θ implies that $\Theta = \widehat{\Theta}$. For this reason, we will use the term **path-consistent set** as a synonym for a set that is the reduced closure of itself. Finally, computing $\widehat{\Theta}$ is polynomial in the size of Θ [14; 15].

7.3 The ORD-Horn Subclass

Previous results on the tractability of ISAT($\mathcal{S}$) (and hence ISI($\mathcal{S}$)) for some subclass $\mathcal{S} \subseteq \mathcal{A}$ made use of the *expressibility* of interval formulas over $\mathcal{S}$ as certain logical formulas involving endpoint relations.

As usual, by a **clause** we mean a disjunction of literals, where a **literal** in turn is an atomic formula or a negated atomic formula. As **atomic formulas** we allow $a \leq b$ and $a = b$, where a and b denote endpoints of intervals. The negation of $a = b$ is also written as $a \neq b$. Finite sets of such clause will be denoted by Ω. In the following, we consider a slightly restricted form of clauses, which we call **ORD clauses**. These clauses do not contain negations of atoms of the form $(a \leq b)$, i.e., they only contain literals of the form:

$$a = b,\ a \leq b,\ a \neq b.$$

The **ORD-clause form** of an interval formula ϕ, written $\pi(\phi)$, is the set of ORD clauses over endpoint relations that is equivalent to ϕ, i.e., every interval model of ϕ can be transformed into a model of the ORD-clause form over the reals and *vice versa* using the obvious transformation. Consider, for instance, $\pi(X\ \{\mathsf{d}, \mathsf{o}, \mathsf{s}\}\ Y)$:

$$\Big\{\begin{array}{llll}(X^- \leq X^+), & (X^- \neq X^+), & (Y^- \leq Y^+), & (Y^- \neq Y^+),\\ (X^- \leq Y^+), & (X^- \neq Y^+), & (Y^- \leq X^+), & (X^+ \neq Y^-),\\ (X^+ \leq Y^+), & (X^+ \neq Y^+)\end{array}\Big\}.$$

The function $\pi(\cdot)$ is extended to finite sets of interval formulas in the obvious way, i.e., for identical intervals in Θ, identical endpoints are used in $\pi(\Theta)$. Similarly to the notions of I-satisfiability, we define R-**satisfiability** of Ω to be the satisfiability of Ω over the real numbers.

Proposition 2 *Θ is I-satisfiable iff $\pi(\Theta)$ is R-satisfiable.*

Not all relations permit a ORD-clause form that is as concise as the the one shown above, which contains only *unit clauses*. However, in particular those relations that allow for such a clause form have interesting computational properties. For instance, the **continuous endpoint subclass** (which is denoted by $\mathcal{C}$) can be defined as the subclass of interval relations that (1) permit a clause form that contains only unit clauses, and (2) for each unit clause $a \neq b$, the clause form contains also a unit clause of the form $a \leq b$ or $b \leq a$.

As demonstrated above, the relation $\{\mathsf{d}, \mathsf{o}, \mathsf{s}\}$ is a member of the continuous endpoint subclass. This subclass has the favorable property that the path-consistency method solves ISI($\mathcal{C}$) [20; 22; 24]. A slight generalization of the continuous endpoint subclass is the **pointisable subclass** (denoted by $\mathcal{P}$) that is defined in the same way as $\mathcal{C}$, but without condition (2). Path-consistency is not sufficient for solving ISI($\mathcal{P}$) [20] but still sufficient for deciding satisfiability [11; 23].

We generalize this approach by being more liberal concerning the clause form. We consider the subclass of Allen's interval algebra such that the relations permit an ORD-clause form containing only clauses with *at most one positive literal*, which we call **ORD-Horn clauses**. The subclass defined in this way is called **ORD-Horn subclass**, and we use the symbol $\mathcal{H}$ to refer to it. The relation $\{\mathsf{o}, \mathsf{s}, \mathsf{f}^{\smile}\}$ is, for instance, an element of $\mathcal{H}$ because $\pi(X \{\mathsf{o}, \mathsf{s}, \mathsf{f}^{\smile}\} Y)$ can be expressed as follows:

$$\Big\{\begin{array}{llll}(X^- \leq X^+), & (X^- \neq X^+), & (Y^- \leq Y^+), & (Y^- \neq Y^+),\\ (X^- \leq Y^-), & (X^- \leq Y^+), & (X^- \neq Y^+), & \\ (Y^- \leq X^+), & (X^+ \neq Y^-), & (X^+ \leq Y^+), & \\ (X^- \neq Y^- \vee X^+ \neq Y^+) & & & \end{array}\Big\}.$$

By definition, the ORD-Horn subclass contains the pointisable subclass. Further, by the above example, this inclusion is strict.

Consider now the theory *ORD* that axiomatizes "=" as an equivalence relation and "$\leq$" as a partial ordering over the equivalence classes:

$$\begin{array}{lllll}\forall x, y: & x \leq y \wedge y \leq z & \rightarrow & x \leq z & \text{(Transitivity)}\\ \forall x: & x \leq x & & & \text{(Reflexivity)}\\ \forall x, y: & x \leq y \wedge y \leq x & \rightarrow & x = y & \text{(Antisymmetry)}\\ \forall x, y: & x = y & \rightarrow & x \leq y & \\ \forall x, y: & x = y & \rightarrow & y \leq x. & \end{array}$$

Although this theory is much weaker, and hence allows for more models than the intended models of sets of ORD clauses, R-satisfiability of a finite set Ω of ORD clauses is nevertheless equivalent to the satisfiability of $\Omega \cup \mathit{ORD}$ over arbitrary interpretations.

Proposition 3 *A finite set of ORD clauses Ω is R-satisfiable iff $\Omega \cup \mathit{ORD}$ is satisfiable.*

Note that *ORD* is a *Horn theory*, i.e., a theory containing only Horn clauses. Since the ORD-clause form of interval formulas over $\mathcal{H}$ is also Horn, tractability of ISAT($\mathcal{H}$) would follow, provided we could replace *ORD* by a propositional Horn theory. In order to decide satisfiability of a set of ORD clauses Ω in *ORD*, however, we can restrict ourselves to Herbrand interpretations, i.e, interpretations that have only the endpoints of all intervals mentioned in Ω as objects. In the following, ORD_Ω shall denote the axioms of *ORD* instantiated to all endpoints mentioned in Ω. As a specialization of the Herbrand theorem, we obtain the next proposition.

Proposition 4 *$\Omega \cup \mathit{ORD}$ is satisfiable iff $\Omega \cup \mathit{ORD}_\Omega$ is satisfiable.*

From that, polynomiality of ISAT($\mathcal{H}$) is immediate.

Theorem 5 ISAT($\mathcal{H}$) *is polynomial.*

7.4 The Applicability of Path-Consistency

Enumerating the ORD-Horn subclass reveals that there are 868 relations (including the null relation $\perp$) in Allen's interval algebra that can be expressed using ORD-Horn clauses. Since the full algebra contains $2^{13} = 8192$ relations, $\mathcal{H}$ covers more than 10% of the full algebra. Comparing this with the continuous endpoint subclass $\mathcal{C}$, which contains 83 relations, and the pointisable subclass $\mathcal{P}$, which contains 188 relations,[6] having shown tractability for $\mathcal{H}$ is a clear improvement over previous results. However, there remains the question of whether the "traditional" method of reasoning in Allen's interval algebra, i.e., constraint propagation, gives reasonable results. As it turns out, this is indeed the case.

Theorem 6 *Let $\widehat{\Theta}$ be a path-consistent set of interval formulas over $\mathcal{H}$. Then $\widehat{\Theta}$ is I-satisfiable iff $(X \perp Y) \notin \widehat{\Theta}$.*

Proof Sketch. A case analysis over the possible non-unit clauses in $\pi(\widehat{\Theta}) \cup ORD_{\pi(\widehat{\Theta})}$ reveals that no new units can be derived by *positive unit resolution*, if the ORD-clause form of the interval formulas satisfies the requirement that it contains all implied atoms and the clauses are minimal. By refutation completeness of positive unit resolution [9], the claim follows. ■

The only remaining part we have to show is that transforming Θ over $\mathcal{H}$ into its equivalent path-consistent form $\widehat{\Theta}$ does not result in a set that contains relations not in $\mathcal{H}$. In order to show this we prove that $\mathcal{H}$ is closed under converse, intersection, and composition, i.e., $\mathcal{H}$ (together with these operations) defines a **subalgebra** of Allen's interval algebra.

Theorem 7 *$\mathcal{H}$ is closed under converse, intersection, and composition.*

Proof Sketch. The main problem is to show that the composition of two relations has an ORD-Horn form. We show that by proving that any minimal clause C implied by $\pi(\{XRY, YSZ\})$ is either ORD-Horn or there exists a set of ORD-Horn clauses that are implied by $\pi(\{XRY, YSZ\})$ and imply C. ■

From that it follows straightforwardly that ISAT($\mathcal{H}$) is decided by the path-consistency method.

Theorem 8 *If Θ is a set over $\mathcal{H}$, then Θ is satisfiable iff $(X \perp Y) \notin \widehat{\Theta}$ for all intervals X, Y.*

7.5 The Borderline between Tractable and NP-complete Subclasses

Having identified the tractable fragment $\mathcal{H}$ that contains the previously identified tractable fragment $\mathcal{P}$ and that is considerably larger than $\mathcal{P}$ is satisfying in itself. However, such a result also raises the question for the the boundary between polynomiality and NP-completeness in Allen's interval algebra.

While the introduction of the algebraic structure on the set of expressible interval relations may have seem to be only motivated by the particular approximation algorithm employed, this structure is also useful when we explore the computational properties of restricted

[6] An enumeration of $\mathcal{C}$ and $\mathcal{P}$ is given by van Beek and Cohen [22].

problems. For any arbitrary subset $\mathcal{S} \subseteq \mathcal{A}$, $\overline{\mathcal{S}}$ shall denote the **closure** of $\mathcal{S}$ under converse, intersection, and composition. In other words, $\overline{\mathcal{S}}$ is the carrier of the **least subalgebra generated by** $\mathcal{S}$.

Theorem 9 ISAT($\overline{\mathcal{S}}$) *can be polynomially transformed to* ISAT($\mathcal{S}$).

Proof Sketch. The main things to note are that (1) if $R = S^{\smile}$ and $S \in \mathcal{S}$ then the interval formula (XRY) in Θ can be replaced by (YSX); (2) if $R = S \cap T$ and $S, T \in \mathcal{S}$, then (XRY) in Θ can be replaced by the two formulas (XSY), (XTY); (3) if $R = S \circ T$ and $S, T \in \mathcal{S}$, then (XRY) in Θ can be replaced by (XSZ), (ZTY), where Z is a fresh interval. Clearly, if Θ is I-satisfiable then the modified set is and *vice versa*. ∎

In other words, once we have proven that satisfiability is polynomial for some set $\mathcal{S} \subseteq \mathcal{A}$, this result extends to the least subalgebra generated by $\mathcal{S}$. Conversely, NP-hardness for a subalgebra is "inherited" by all subsets that generate this subalgebra.

Corollary 10 ISAT($\mathcal{S}$) *is polynomial iff* ISAT($\overline{\mathcal{S}}$) *is polynomial and* ISAT($\mathcal{S}$) *is NP-complete iff* ISAT($\overline{\mathcal{S}}$) *is NP-complete.*

It still takes some effort to prove that a given fragment $\mathcal{S}$ is a *maximal* tractable subclass of Allen's interval algebra. Firstly, one has to show that $\mathcal{S} = \overline{\mathcal{S}}$. For the ORD-Horn subclass, this has been done in Theorem 7. Secondly, one has to show that ISAT($\mathcal{T}$) is NP-complete for all *minimal* subalgebras $\mathcal{T}$ that strictly contain $\mathcal{S}$. This, however, means that these subalgebras have to be identified. Certainly, such a case analysis cannot be done manually. In fact, we used a program to identify the minimal subalgebras strictly containing $\mathcal{H}$. An analysis of the clause form of the relations appearing in these subalgebras leads us to the formulation of the following machine-verifiable lemma.

Lemma 11 *Let* $\mathcal{S} \subseteq \mathcal{A}$ *be any set of interval relations that strictly contains* $\mathcal{H}$. *Then* $\{\mathsf{d}, \mathsf{d}^{\smile}, \mathsf{o}^{\smile}, \mathsf{s}^{\smile}, \mathsf{f}\}$ *or* $\{\mathsf{d}^{\smile}, \mathsf{o}, \mathsf{o}^{\smile}, \mathsf{s}^{\smile}, \mathsf{f}^{\smile}\}$ *is an element of* $\overline{\mathcal{S}}$.

For reasons of simplicity, we will not use the ORD clause form in the following, but a clause form that also contains literals over the relations $\geq, <, >$. Then the clause form for the relations mentioned in the lemma can be given as follows:

$$\begin{aligned}
\pi(X\,\{\mathsf{d}, \mathsf{d}^{\smile}, \mathsf{o}^{\smile}, \mathsf{s}^{\smile}, \mathsf{f}\}\,Y) &= \Big\{(X^- < X^+), \quad (Y^- < Y^+), \\
&\qquad (X^- < Y^+), \quad (X^+ > Y^-), \\
&\qquad \Big((X^- > Y^-) \vee (X^+ > Y^+)\Big)\Big\}, \\
\pi(X\,\{\mathsf{d}^{\smile}, \mathsf{o}, \mathsf{o}^{\smile}, \mathsf{s}^{\smile}, \mathsf{f}^{\smile}\}\,Y) &= \Big\{(X^- < X^+), \quad (Y^- < Y^+), \\
&\qquad (X^- < Y^+), \quad (X^+ > Y^-), \\
&\qquad \Big((X^- < Y^-) \vee (X^+ > Y^+)\Big)\Big\}.
\end{aligned}$$

We will show that each of these relations together with the two relations $\{\prec, \mathsf{d}^{\smile}, \mathsf{o}, \mathsf{m}, \mathsf{f}^{\smile}\}$ and $\{\prec, \mathsf{d}, \mathsf{o}, \mathsf{m}, \mathsf{s}\}$, which are elements of $\mathcal{C}$, are enough for making the interval satisfiability problem NP-complete. The clause form of these relations looks as follows:

$$\begin{aligned}
\pi(X\,\{\prec, \mathsf{d}^{\smile}, \mathsf{o}, \mathsf{m}, \mathsf{f}^{\smile}\}\,Y) &= \Big\{(X^- < X^+), \quad (Y^- < Y^+), \\
&\qquad (X^- < Y^-), \quad (X^- < Y^+)\Big\} \\
\pi(X\,\{\prec, \mathsf{d}, \mathsf{o}, \mathsf{m}, \mathsf{s}\}\,Y) &= \Big\{(X^- < X^+), \quad (Y^- < Y^+), \\
&\qquad (X^+ < Y^+), \quad (X^- < Y^+)\Big\}
\end{aligned}$$

Lemma 12 ISAT($\mathcal{S}$) *is NP-complete if*

1. $\mathcal{N}_1 = \left\{\{\prec, \mathsf{d}^\smile, \mathsf{o}, \mathsf{m}, \mathsf{f}^\smile\}, \{\prec, \mathsf{d}, \mathsf{o}, \mathsf{m}, \mathsf{s}\}, \{\mathsf{d}, \mathsf{d}^\smile, \mathsf{o}^\smile, \mathsf{s}^\smile, \mathsf{f}\}\right\} \subseteq \mathcal{S}$, *or*
2. $\mathcal{N}_2 = \left\{\{\prec, \mathsf{d}^\smile, \mathsf{o}, \mathsf{m}, \mathsf{f}^\smile\}, \{\prec, \mathsf{d}, \mathsf{o}, \mathsf{m}, \mathsf{s}\}, \{\mathsf{d}^\smile, \mathsf{o}, \mathsf{o}^\smile, \mathsf{s}^\smile, \mathsf{f}^\smile\}\right\} \subseteq \mathcal{S}$.

Proof Sketch. Since ISAT($\mathcal{A}$) $\in$ NP, membership in NP follows.

For the NP-hardness part we will show that 3SAT can be polynomially transformed to ISAT($\mathcal{N}_k$). We will first prove the claim for $\mathcal{N}_1$. Let $D = \{C_i\}$ be a set of clauses, where $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ and the $l_{i,j}$'s are literal occurrences. We will construct a set of interval formulas Θ over $\mathcal{N}_1$ such that Θ is I-satisfiable iff D is satisfiable.

For each literal occurrence $l_{i,j}$ a pair of intervals $X_{i,j}$ and $Y_{i,j}$ is introduced, and the following first group of interval formulas is put into Θ:

$$(X_{i,j}\ \{\mathsf{d}, \mathsf{d}^\smile, \mathsf{o}^\smile, \mathsf{s}^\smile, \mathsf{f}\}\ Y_{i,j}).$$

This implies that $\pi(\Theta)$ contains among other things the following clauses $(X^-_{i,j} > Y^-_{i,j} \vee X^+_{i,j} > Y^+_{i,j})$.

Additionally, we add a second group of formulas for each clause C_i:

$$\begin{array}{l}(X_{i,2}\ \{\prec, \mathsf{d}^\smile, \mathsf{o}, \mathsf{m}, \mathsf{f}^\smile\}\ Y_{i,1}),\\ (X_{i,3}\ \{\prec, \mathsf{d}^\smile, \mathsf{o}, \mathsf{m}, \mathsf{f}^\smile\}\ Y_{i,2}),\\ (X_{i,1}\ \{\prec, \mathsf{d}^\smile, \mathsf{o}, \mathsf{m}, \mathsf{f}^\smile\}\ Y_{i,3}),\end{array}$$

which leads to the inclusion of the clauses $(Y^-_{i,1} > X^-_{i,2})$, $(Y^-_{i,2} > X^-_{i,3})$, $(Y^-_{i,3} > X^-_{i,1})$ in $\pi(\Theta)$.

This construction leads to the situation that there is no model of Θ that satisfies for given i all disjuncts of the form $(X^-_{i,j} > Y^-_{i,j})$ in the clause form of $\pi(X_{i,j}\{\mathsf{d}, \mathsf{d}^\smile, \mathsf{o}^\smile, \mathsf{s}^\smile, \mathsf{f}\}Y_{i,j})$. If the jth disjunct $(X^-_{i,j} > Y^-_{i,j})$ is unsatisfied in an I-model of Θ, we will interpret this as the satisfaction of the literal occurrence $l_{i,j}$ in C_i of D.

In order to guarantee that if a literal occurrence $l_{i,j}$ is interpreted as satisfied, then all complementary literal occurrences in D are interpreted as unsatisfied, the following third group of interval formulas for complementary literal occurrences $l_{i,j}$ and $l_{g,h}$ are added to Θ:

$$(X_{g,h}\ \{\prec, \mathsf{d}, \mathsf{o}, \mathsf{m}, \mathsf{s}\}\ Y_{i,j}), \quad (X_{i,j}\ \{\prec, \mathsf{d}, \mathsf{o}, \mathsf{m}, \mathsf{s}\}\ Y_{g,h}),$$

which leads to the inclusion of $(Y^+_{i,j} > X^+_{g,h})$, $(Y^+_{g,h} > X^+_{i,j})$. This construction guarantees that Θ is I-satisfiable iff D is satisfiable.

The transformation for $\mathcal{N}_2$ is similar. ∎

Based on this result, it follows straightforwardly that $\mathcal{H}$ is indeed a maximal tractable subclass of $\mathcal{A}$.

Theorem 13 *If $\mathcal{S}$ strictly contains $\mathcal{H}$, then* ISAT($\mathcal{S}$) *is NP-complete.*

The next question is whether there are other maximal tractable subclasses that are incomparable with $\mathcal{H}$. One example of an incomparable tractable subclass is $\mathcal{U} = \{\{\prec, \succ\}, \top\}$. Since $\{\prec, \succ\}$ has no ORD-Horn clause form, this subclass is incomparable with $\mathcal{H}$, and since all sets of interval formulas over $\mathcal{U}$ are trivially satisfiable (by making all intervals disjoint), ISAT($\mathcal{U}$) can be decided in constant time. The subclass $\mathcal{U}$ is, of course, not a very

interesting fragment. Provided we are interested in temporal reasoning in the framework as described by Allen [2], one necessary requirement is that *all basic relations* are contained in the subclass. A machine-assisted exploration of the space of subalgebras leads us to the following machine-verifiable lemma.

Lemma 14 *If $\mathcal{S}$ is a subclass that contains the thirteen basic relations, then $\overline{\mathcal{S}} \subseteq \mathcal{H}$, or $\{\mathsf{d}, \mathsf{d}^\smile, \mathsf{o}^\smile, \mathsf{s}^\smile, \mathsf{f}\}$ or $\{\mathsf{d}^\smile, \mathsf{o}, \mathsf{o}^\smile, \mathsf{s}^\smile, \mathsf{f}^\smile\}$ is an element of $\overline{\mathcal{S}}$.*

Using the fact that $\{\prec, \mathsf{d}^\smile, \mathsf{o}, \mathsf{m}, \mathsf{f}^\smile\}, \{\prec, \mathsf{d}, \mathsf{o}, \mathsf{m}, \mathsf{s}\}$ are elements of the least subalgebra generated by the set of basic relations and employing Lemma 12 again, we obtain the quite satisfying result that $\mathcal{H}$ is in fact the unique greatest tractable subclass amongst the subclasses containing all basic relations.

Theorem 15 *Let $\mathcal{S}$ be any subclass of $\mathcal{A}$ that contains all basic relations. Then either $\mathcal{S} \subseteq \mathcal{H}$ and* ISAT($\mathcal{S}$) *is polynomial or* ISAT($\mathcal{S}$) *is NP-complete.*

7.6 Conclusion

We have identified a new tractable subclass of Allen's interval algebra, which we call *ORD-Horn subclass* and which contains the previously identified *continuous endpoint* and *pointisable* subclasses. Enumerating the ORD-Horn subclass reveals that this subclass contains 868 elements out of 8192 elements in the full algebra, i.e., more than 10% of the full algebra. Comparing this with the continuous endpoint subclass that covers approximately 1% and with the pointisable subclass that covers 2%, our result is a clear improvement in quantitative terms.

Furthermore, we showed that the "traditional" method of reasoning in Allen's interval algebra, namely, the *path-consistency method*, is sufficient for deciding satisfiability in the ORD-Horn subclass. In other words, our results indicate that the path-consistency method has a much larger range of applicability for reasoning in Allen's interval algebra than previously believed—if we are mainly interested in satisfiability.

Provided that a restriction to the subclass $\mathcal{H}$ is not possible in an application, our results may be employed in designing faster backtracking algorithms for the full algebra [19; 21]. Since our subclass contains significantly more relations than other tractable subclasses, the branching factor in a backtrack search can be considerably decreased if the ORD-Horn subclass is used.

Finally, we showed that it is impossible to improve on our results. Using a machine-generated case analysis, we showed that the ORD-Horn subclass is a *maximal* tractable subclass of Allen's interval algebra and, in fact, even the *unique greatest* tractable subclass in the set of subclasses that contain all basic relations. In other words, the ORD-Horn subclass presents an optimal tradeoff between expressiveness and tractability [12] for reasoning in Allen's interval algebra.

Acknowledgements: We would like to thank Peter Ladkin, Henry Kautz, Ron Shamir, Bart Selman, and Marc Vilain for discussions concerning the topics of this paper. In particular, Ron corrected an overly strong claim we made. In addition, we would like to thank Christer Bäckström for comments on an earlier version of this paper.

7.7 References

[1] *Proceedings of the 6th National Conference of the American Association for Artificial Intelligence*, Seattle, WA, July 1987.

[2] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, Nov. 1983.

[3] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[4] J. F. Allen. Temporal reasoning and planning. In J. F. Allen, H. A. Kautz, R. N. Pelavin, and J. D. Tenenberg, editors, *Reasoning about Plans*, chapter 1, pages 1–67. Morgan Kaufmann, San Mateo, CA, 1991.

[5] J. F. Allen and P. J. Hayes. A common-sense theory of time. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 528–531, Los Angeles, CA, Aug. 1985.

[6] J. F. Allen and J. A. Koomen. Planning using a temporal world model. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 741–747, Karlsruhe, Germany, Aug. 1983.

[7] S. K. Feiner, D. J. Litman, K. R. McKeown, and R. J. Passonneau. Towards coordinated temporal multimedia presentation. In M. Maybury, editor, *Intelligent Multi Media*. AAAI Press, Menlo Park, CA, 1993. Forthcoming.

[8] M. C. Golumbic and R. Shamir. Algorithms and complexity for reasoning about time. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 741–747. MIT Press, San Jose, CA, July 1992.

[9] L. Henschen and L. Wos. Unit refutations and Horn sets. *Journal of the Association for Computing Machinery*, 21:590–605, 1974.

[10] P. B. Ladkin. Models of axioms for time intervals. In AAAI-87 [1], pages 234–239.

[11] P. B. Ladkin and R. Maddux. On binary constraint networks. Technical report, Kestrel Institute, Palo Alto, 1988.

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

[13] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[14] A. K. Mackworth and E. C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–73, 1985.

[15] U. Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Science*, 7:95–132, 1974.

[16] B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. DFKI Research Report RR-93-11, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany, Mar. 1993.

[17] K. Nökel. Convex relations between time intervals. In J. Rettie and K. Leidlmair, editors, *Proceedings der 5. Österreichischen Artificial Intelligence-Tagung*, pages 298–302. Springer-Verlag, Berlin, Heidelberg, New York, 1989.

[18] F. Song and R. Cohen. The interpretation of temporal relations in narrative. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 745–750, Saint Paul, MI, Aug. 1988.

[19] R. E. Valdéz-Pérez. The satisfiability of temporal constraint networks. In AAAI-87 [1], pages 256–260.

[20] P. van Beek. Approximation algorithms for temporal reasoning. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1291–1296, Detroit, MI, Aug. 1989. Morgan Kaufmann.

[21] P. van Beek. Reasoning about qualitative temporal information. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 728–734, Boston, MA, Aug. 1990. MIT Press.

[22] P. van Beek and R. Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.

[23] M. B. Vilain and H. A. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the 5th National Conference of the American Association for Artificial Intelligence*, pages 377–382, Philadelphia, PA, Aug. 1986.

[24] M. B. Vilain, H. A. Kautz, and P. G. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, San Mateo, CA, 1989.

[25] R. Weida and D. Litman. Terminological reasoning with constraint networks and an application to plan recognition. In B. Nebel, W. Swartout, and C. Rich, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference*, pages 282–293, Cambridge, MA, Oct. 1992. Morgan Kaufmann.

8

$\mathcal{ALCP}$ – Handling Uncertainty in Description Logics

Jochen Heinsohn[1]

ABSTRACT This paper presents the language $\mathcal{ALCP}$ which is a probabilistic extension of terminological logics and aims at closing the gap between *terminological knowledge representation* and *modeling uncertainty*: while classical terminological frameworks exclude the possibility to handle uncertain concept descriptions involving, e.g., "usually true" concept properties, generalized quantifiers, or exceptions, purely numerical approaches for handling uncertain knowledge in general are unable to consider terminological knowledge. The paper concentrates on the formal semantics underlying the language $\mathcal{ALCP}$ and the associated probabilistic model based on classes of probabilities. $\mathcal{ALCP}$ maintains the original performance of terminological languages of drawing inferences in a hierarchy of terminological definitions. It enlarges the range of applicability to real world domains determined not only by descriptive but also by uncertain knowledge. On the basis of the language construct "p-conditioning" it is shown how statistical information on concept dependencies can be represented. To allow for probabilistic inferences and to guarantee (terminological and probabilistic) consistency, several requirements have to be met. In particular, these requirements allow one to infer implicitly existent probabilistic relationships and their quantitative computation. By explicitly introducing restrictions for the ranges derived by instantiating the consistency requirements, anomalies and exceptions can also be handled. Consequently, $\mathcal{ALCP}$ applies to domains where both term descriptions and uncertainty have to be represented.

8.1 Introduction

Research in knowledge representation led to the development of terminological logics [Patel-Schneider *et al.*, 1990] which originated mainly in Brachman's KL-ONE [Brachman and Schmolze, 1985] and are called *description logics* [Patil *et al.*, 1992] since 1991. In such languages the terminological formalism (*TBox*) is used to represent a hierarchy of terms (*concepts*) that are partially ordered by a subsumption relation: concept B is *subsumed by* concept A, if, and only if, 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, KRIS, KRYPTON, MESON, SB-ONE, and YAK (for overview and analyses see [Sigart Bulletin, 1991; Nebel, 1990] and [Heinsohn *et al.*, 1993]) 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 for finding 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 pos-

[1]Project WIP, DFKI Saarbrücken, email: heinsohn@dfki.uni-sb.de

sibility of *defining* concepts (or at least specifying 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 [Brachman, 1985] this may happen in "natural" environments (in contrast to "technical/mathematical" environments). The source of this problem is the fact that in natural environments, besides their description, terms can only be characterized as having additional *typical* properties or properties that are, for instance, *usually* true. If such properties are interpreted as being categorical, this can lead to problems concerning *multiple inheritance*. One example that can be used to highlight these problems is known as the "Nixon diamond": quakers are typically pacifist, republicans are typically non-pacifist, and Nixon is known to be both quaker and republican. Modeling these relationships categorically results in the detection of a *contradiction*. However, in the real world such properties often are only *tendencies*, i.e., republicans "usually" are non-pacifist, for example. Tendencies as well as differences in these tendencies cannot be considered in the framework of term definitions.

We propose an extension of terminological logics that allows one to handle the problems discussed above [Heinsohn, 1993; Heinsohn, 1991; Heinsohn, 1992]. First, we briefly introduce $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1991], a propositionally complete terminological language containing the logical connectives conjunction, disjunction and 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. In Section 8.3 we extend $\mathcal{ALC}$ by defining syntax and semantics of *probabilistic conditioning* (p-conditioning), a construct aimed at considering non-terminological knowledge sources and based on a statistical interpretation. In Section 8.4 we introduce the formal model underlying a terminology and a set of p-conditionings. The classes of probabilities induced by a terminology and a set of p-conditionings are described in Section 8.5. As demonstrated in Section 8.6, on the basis of the terminological and probabilistic knowledge, certain consistency requirements have to be met. Moreover, the associated probabilistic constraints allow one to infer implicitly existent probabilistic relationships and their quantitative computation. Related work and the conclusions are given in Sections 8.7 and 8.8, respectively. While this paper mainly focuses on TBox and statistical aspects, the consideration of an ABox would mean the ability to draw inferences about "probabilistic memberships" of instances [Heinsohn, 1993].

8.2 The Terminological Formalism

The basic elements of the terminological language $\mathcal{ALC}$ [Schmidt-Schauß and Smolka, 1991] are concepts and roles (denoting subsets of the domain of interest and binary relations over this domain, respectively). Assume that $\top$ ("top", denoting the entire domain) and $\bot$ ("bottom", denoting the empty set) are concept symbols, that A denotes a concept symbol, and R denotes a role. Then the concepts (denoted by letters C and D) of the language $\mathcal{ALC}$ are built according to the abstract syntax rule

$$\begin{aligned} C, D \rightarrow \quad & A \mid \top \mid \bot \\ \mid \quad & C \sqcap D \\ \mid \quad & C \sqcup D \\ \mid \quad & \neg C \\ \mid \quad & \forall R{:}C \\ \mid \quad & \exists R{:}C \end{aligned} \quad .$$

To introduce a formal semantics of $\mathcal{ALC}$ we give a translation into set theoretical expressions with $\mathcal{D}$ being the domain of discourse. For that purpose, we define a mapping $\mathcal{E}$ that maps every concept description to a subset of $\mathcal{D}$ and every role to a subset of $\mathcal{D} \times \mathcal{D}$ in the following way:

$$\begin{aligned}
\mathcal{E}[\![\top]\!] &= \mathcal{D} \\
\mathcal{E}[\![\bot]\!] &= \emptyset \\
\mathcal{E}[\![(C \sqcap D)]\!] &= \mathcal{E}[\![C]\!] \cap \mathcal{E}[\![D]\!] \\
\mathcal{E}[\![(C \sqcup D)]\!] &= \mathcal{E}[\![C]\!] \cup \mathcal{E}[\![D]\!] \\
\mathcal{E}[\![(\neg C)]\!] &= \mathcal{D} \setminus \mathcal{E}[\![C]\!] \\
\mathcal{E}[\![(\forall R : C)]\!] &= \{x \in \mathcal{D} | \text{ for all } y \in \mathcal{D} : ((x,y) \in \mathcal{E}[\![R]\!] \Rightarrow y \in \mathcal{E}[\![C]\!])\} \\
\mathcal{E}[\![(\exists R : C)]\!] &= \{x \in \mathcal{D} | \text{ there exists } y \in \mathcal{D} : ((x,y) \in \mathcal{E}[\![R]\!] \wedge y \in \mathcal{E}[\![C]\!])\}
\end{aligned}$$

Concept descriptions are used to state necessary, or necessary and sufficient conditions by means of specializations "$\sqsubseteq$" or definitions "$\doteq$", respectively. Assuming symbol A and concept description C, then "$A \sqsubseteq C$" means the inequality $\mathcal{E}[\![A]\!] \subseteq \mathcal{E}[\![C]\!]$, and "$A \doteq C$" means the equation $\mathcal{E}[\![A]\!] = \mathcal{E}[\![C]\!]$. A set of well formed concept definitions and specializations forms a *terminology*, if every concept symbol appears at most once on the left hand side and there are no terminological cycles [Nebel, 1991]. A concept C_1 is said to be *subsumed by* a concept C_2 in a terminology $\mathcal{T}$, written $C_1 \preceq_{\mathcal{T}} C_2$, iff the inequality $\mathcal{E}[\![C_1]\!] \subseteq \mathcal{E}[\![C_2]\!]$ holds for all extension functions satisfying the equations introduced in $\mathcal{T}$. $\mathcal{ALC}$ is used in the implemented prototype system KRIS [Baader and Hollunder, 1991] where all reasoning facilities such as computing the subsumption hierarchy are realized by sound and complete algorithms.

Terminological languages as $\mathcal{ALC}$ can be usefully applied to definitional world knowledge. For instance, we may introduce

Example 8.2.1

$$\begin{aligned}
\mathit{animal} &\sqsubseteq \top \\
\mathit{flying} &\sqsubseteq \top \\
\mathit{antarctic_animal} &\sqsubseteq \mathit{animal} \\
\mathit{bird} &\doteq \mathit{animal} \sqcap (\forall \mathit{moves_by} : \mathit{flying}) \\
\mathit{antarctic_bird} &\doteq \mathit{antarctic_animal} \sqcap \mathit{bird} \\
\mathit{penguin} &\sqsubseteq \mathit{antarctic_bird}
\end{aligned}$$

Definition 1 *Let $\mathcal{T}$ be a terminology. The set*

$$\mathrm{mod}(\mathcal{T}) \stackrel{\mathrm{def}}{=} \{\mathcal{E} | \mathcal{E} \textit{ extension function of } \mathcal{T}\} \tag{8.1}$$

is called set of models of $\mathcal{T}$.

To characterize the expressiveness of terminological languages, we will examine the three different relations imaginable between two concept *extensions*, i.e., (i) inclusion, (ii) disjointness, and (iii) overlapping:

$$\text{(i) } \mathcal{E}[\![C_1]\!] \subseteq \mathcal{E}[\![C_2]\!], \quad \text{(ii) } \mathcal{E}[\![C_1]\!] \cap \mathcal{E}[\![C_2]\!] = \emptyset, \quad \text{(iii) } \mathcal{E}[\![C_1]\!] \cap \mathcal{E}[\![C_2]\!] \neq \emptyset \tag{8.2}$$

The first case *can* be caused by (terminological) subsumption. To express extensional inclusion (i) *without* a subsumption relation on terms, some hybrid systems introduced non-terminological language constructs such as `implication` [MacGregor and Bates, 1987; Owsnicki-Klewe, 1988] or `assert-rule` [Borgida *et al.*, 1989]. Disjointness (ii) *can* be a terminological

property. This is the case if, for instance, the above language construct "concept negation" as contained in the expression $C_1 \sqsubseteq C, C_2 \doteq (C \sqcap \neg C_1)$ is used. To express non-terminological disjointness between concepts, some systems use the language construct `disjoint`.

However, the information given in case (iii) cannot be reasonably used in existing terminological logics. It seems to be more suitable to generally consider the "degree of intersection" between the respective concept's extensions and to characterize it using an appropriate technique. The idea behind this generalization is to use a probabilistic semantics.

8.3 The Probabilistic Conditioning

In the following we consider only one representative for equivalent concept expressions (such as A, $A \sqcap \top$, $A \sqcap A$). The algebra based on representatives of equivalence classes and on the logical connectives $\sqcap$, $\sqcup$, and $\neg$ is known as *Lindenbaum algebra* of the set $\mathcal{S}$ of concept symbols. We use the symbol $\mathcal{C}$ for the set of concept descriptions. $\mathcal{D}$ is assumed to be finite. As a language construct that takes into account *all* cases (8.2), we introduce the notion of *p-conditioning* which can be viewed as a generalization of the above mentioned implication construct: the language construct $C_1 \overset{[p_l,p_u]}{\rightarrow} C_2$ is called p-conditioning, iff $[p_l, p_u]$ is a subrange of real numbers with $0 \leq p_l \leq p_u \leq 1$, and $C_1, C_2 \in \mathcal{C}$ holds. The semantics is defined as follows:

Definition 2 *An extension function $\mathcal{E}$ over $\mathcal{C}$ satisfies a p-conditioning $C_1 \overset{[p_l,p_u]}{\rightarrow} C_2$, written $\models_{\mathcal{E}} C_1 \overset{[p_l,p_u]}{\rightarrow} C_2$, iff*

$$\frac{|\mathcal{E}[\![C_1 \sqcap C_2]\!]|}{|\mathcal{E}[\![C_1]\!]|} \quad \in \quad [p_l, p_u] \tag{8.3}$$

holds for concepts $C_1, C_2 \in \mathcal{C}$, $\mathcal{E}[\![C_1]\!] \neq \emptyset$.

From the above it is obvious that we use the *relative cardinality* for interpreting the notion of p-conditioning. For illustrating the meaning of Definition 2, assume that an observer examines the flying ability of a real class of birds. When finishing his study he may have learned that, different from the model of Example 8.2.1, relation *moves_by:flying* holds only for a certain percentage of the birds. The notion of p-conditioning now allows a representation of universal knowledge of statistical kind in a way that maintains the semantics of the roles: the new concept *flying_object* is created with role *moves_by* restricted to range *flying*. The uncertainty is represented by a p-conditioning stating that "at least 95% of *bird*s are *flying_object*s that, by definition, all move by flying". The now more detailed view to the example world leads to the following revision of Example 8.2.1:

Example 8.3.1 *(revises Example 8.2.1)*

$$\begin{array}{rcl}
\mathit{animal} & \sqsubseteq & \top \\
\mathit{flying} & \sqsubseteq & \top \\
\mathit{antarctic_animal} & \sqsubseteq & \mathit{animal} \\
\mathit{bird} & \sqsubseteq & \mathit{animal} \\
\mathit{antarctic_bird} & \doteq & \mathit{antarctic_animal} \sqcap \mathit{bird} \\
\mathit{penguin} & \sqsubseteq & \mathit{antarctic_bird} \\
\mathit{flying_object} & \doteq & \forall \mathit{moves_by} : \mathit{flying} \\
\mathit{bird} & \overset{[0.95,1]}{\rightarrow} & \mathit{flying_object}
\end{array}$$

This demonstrates that set theory is sufficient for a consistent semantic basis on which both terminological and probabilistic language constructs can be interpreted. On this basis, the p-conditioning serves as a generalization of both the "implication" and the "disjoint" construct (now appearing as $A \overset{1}{\rightarrow} B$ and $A \overset{0}{\rightarrow} B$, respectively) used in many hybrid systems.

8.4 The Formal Model

It is obvious that the set of models of a terminology (see equation (8.1)) is successively restricted if p-conditionings are introduced.

In case of a concrete application domain, on the basis of p-conditionings (uncertain) relationships can be described only between some special concepts of a terminology—neither directly representable statistical knowledge nor textbook knowledge that can be indirectly modeled is complete in this sense. Consequently, the question arises in which way, starting with a set of models restricted wrt. a terminology and a set of p-conditionings, one can *indirectly* derive (uncertain) relationships between other pairs of concepts for which p-conditionings are not explicitly introduced. Below we give an answer to this question by defining the sets of *entailed* and *minimal* p-conditionings. The probabilistic constraints associated with such a formal model are formulated in Section 8.6.

Definition 3 *Let $\mathcal{T}$ be a terminology and $\mathcal{I}$ be a set of p-conditionings. The set*

$$\mathrm{mod}_{\mathcal{T}}(\mathcal{I}) \overset{\mathrm{def}}{=} \{\mathcal{E} \mid \models_{\mathcal{E}} \mathcal{I}\} \cap \mathrm{mod}(\mathcal{T}) \tag{8.4}$$

is called the set of models of $\mathcal{T}$ and $\mathcal{I}$. The set

$$\mathrm{Th}_{\mathcal{T}}(\mathcal{I}) \overset{\mathrm{def}}{=} \{I \mid \textit{for all } \mathcal{E} \in \mathrm{mod}_{\mathcal{T}}(\mathcal{I}) : \models_{\mathcal{E}} I\} \tag{8.5}$$

is called the set of entailed p-conditionings wrt. $\mathcal{I}$ and $\mathcal{T}$. In case of existence

$$\mathrm{min}_{\mathcal{T}}(\mathcal{I}) \overset{\mathrm{def}}{=} \bigcup_{C,D \in \mathcal{C}} \{C \overset{R_{\min}}{\rightarrow} D \mid R_{\min} = \bigcap_{R\,:\,C \overset{R}{\rightarrow} D \in \mathrm{Th}_{\mathcal{T}}(\mathcal{I})} R\} \tag{8.6}$$

is called the set of minimal p-conditionings wrt. $\mathcal{I}$ and $\mathcal{T}$.

These definitions—especially the set defined with (8.6)—describe a formal model that allows an implicit derivation of p-conditionings not explicitly introduced or a further refinement of already known p-conditionings. A set $\mathcal{I}$ of p-conditionings is called *consistent wrt.* $\mathcal{T}$, iff $\mathrm{mod}_{\mathcal{T}}(\mathcal{I}) \neq \emptyset$ holds. In direct relation to (8.6) the question arises whether the minimal sets $R_{\min}$ of real numbers are ranges as it is the case for p-conditionings introduced explicitly.

Proposition 16 *The sets $R_{\min}$ of real numbers defined in (8.6) and associated with minimal p-conditionings form ranges.*

The proof of this proposition is mainly based on the property of convexity of probability classes that are induced by terminological axioms and p-conditionings over the set of atomic concept expressions and are discussed in the following section.

8.5 Induced Sets of Probabilities

In addition to the symbol $\mathcal{C}$ for the set of concept descriptions we use $\mathcal{C}^{\mathrm{A}}$ for the set of *atomic concept expressions* (i.e., the atoms of the Lindenbaum algebra). Atomic concept expressions are of the form $B_1 \sqcap B_2 \sqcap \ldots \sqcap B_m$, where B_i is either a concept symbol A or the negation $\neg A$ of a symbol. The relation $\mathcal{C}^{\mathrm{A}} \subseteq \mathcal{C}$ holds.

A first simple observation is that for every extension function $\mathcal{E} \in \mathrm{mod}_{\mathcal{T}}(\mathcal{I})$ the set of extensions of the elements in $\mathcal{C}^{\mathrm{A}}$ forms a partition of $\mathcal{D}$. A direct consequence of this observation is that every extension function $\mathcal{E}$ uniquely determines a probability over $\mathcal{C}^{\mathrm{A}}$.

Proposition 17 *Let $\mathcal{T}$ be a terminology and $\mathcal{I}$ be a consistent set of p-conditionings. Further, let $\mathcal{E} \in \mathrm{mod}_{\mathcal{T}}(\mathcal{I})$ be an extension function for which*

$$\models_{\mathcal{E}} \top \xrightarrow{p_i} C_i^-, \quad p_i \stackrel{\mathrm{def}}{=} \frac{|\mathcal{E}[\![C_i^-]\!]|}{|\mathcal{D}|} \quad \text{for all } C_i^- \in \mathcal{C}^{\mathrm{A}} \tag{8.7}$$

holds. Then the real-valued set function $P_{\mathcal{E}}$ defined by

$$P_{\mathcal{E}} : 2^{\mathcal{C}^{\mathrm{A}}} \to [0,1],\ P_{\mathcal{E}}(\{C_i^-\}) \stackrel{\mathrm{def}}{=} p_i,\ C_i^- \in \mathcal{C}^{\mathrm{A}} \tag{8.8}$$

is a probability function over $\mathcal{C}^{\mathrm{A}}$.

Example 8.5.1 *Assume the set $\mathcal{T} = \{A \sqsubseteq \top, B \sqsubseteq \top, C \doteq A \sqcap B\}$ of terminological axioms. From $\mathcal{S} = \{\top, A, B\}$ we obtain $\mathcal{C}^{\mathrm{A}} = \{\neg A \sqcap \neg B, \neg A \sqcap B, A \sqcap B, A \sqcap \neg B\}$. Then, $\mathcal{E}$ and $\mathcal{D}$ with $|\mathcal{D}| = 100, |\mathcal{E}[\![A]\!]| = 40, |\mathcal{E}[\![B]\!]| = 20, |\mathcal{E}[\![A \sqcap B]\!]| = 10$ induce a probability function*

$$\begin{aligned} P_{\mathcal{E}} : \neg A \sqcap \neg B &\mapsto 0.5 \\ \neg A \sqcap B &\mapsto 0.1 \\ A \sqcap B &\mapsto 0.1 \\ A \sqcap \neg B &\mapsto 0.3 \end{aligned}$$

Note that every concept can be represented as a disjunction of atomic concept expressions, i.e., for every concept expression $C \in \mathcal{C}$ there exists a subset $D \subseteq \mathcal{C}^{\mathrm{A}}$ of atoms such that $C = \bigsqcup D$. In this way $P_{\mathcal{E}}$ can be extended to concept expressions. In particular,

$$\begin{aligned} P_{\mathcal{E}}(\top) &= P_{\mathcal{E}}(\textstyle\bigsqcup \mathcal{C}^{\mathrm{A}}) = 1 \\ P_{\mathcal{E}}(C) &\geq 0 \text{ for all } C \in \mathcal{C} \\ P_{\mathcal{E}}(C_i \sqcup C_j) &= P_{\mathcal{E}}(C_i) + P_{\mathcal{E}}(C_j) \text{ if } C_i \sqcap C_j = \bot \end{aligned}$$

hold.

Example 8.5.1 shows that, assuming complete knowledge on domain $\mathcal{D}$ and on the cardinalities involved, a probability function $P_{\mathcal{E}}$ over $\mathcal{C}^{\mathrm{A}}$ is induced by the extension function $\mathcal{E}$. However, it is generally more realistic to assume less complete knowledge and cardinalities that are rather relative. Consequently, the set $\mathrm{mod}_{\mathcal{T}}(\mathcal{I})$ generally contains more than one element, so that a *class of probabilities* is induced by a terminology and a set of p-conditionings. The most general set of all probabilities over $\mathcal{C}^{\mathrm{A}}$ is defined by

$$\mathcal{M} \stackrel{\mathrm{def}}{=} \{(p_1, \ldots, p_n) \in \Re^n \mid p_1 + \ldots + p_n = 1,\ p_i \geq 0 \text{ for all } 1 \leq i \leq n\},$$

with $n = 2^m$. For a unique extension function $\mathcal{E}$ the set $\mathcal{M}$ consists of exactly one point in the n-dimensional space $[0,1]^n$. On the other hand, without any knowledge about a terminology and p-conditionings the set $\mathcal{M}$ characterizes the status of *complete ignorance*.

However, if we do not have any knowledge about one unique extension function but we know about a terminology and a p-conditioning I or a set $\mathcal{I}$ of p-conditionings, then by (8.4) a *set* $\mathrm{mod}_{\mathcal{T}}(\mathcal{I})$ of extension functions and also a set

$$\mathcal{M}_{\mathcal{T},\mathcal{I}} \stackrel{\text{def}}{=} \{(p_1,\ldots,p_n) \in \mathcal{M} \mid \text{ exists } \mathcal{E} \in \mathrm{mod}_{\mathcal{T}}(\mathcal{I}) : p_j = P_{\mathcal{E}}(\{C_j^-\}),\ j = 1,\ldots,n\}$$

of probabilities are defined. $\mathcal{M}_{\mathcal{T},\mathcal{I}}$ corresponds to the set of probabilities in $\mathcal{M}$ that are compatible with $\mathcal{T}$ and all p-conditionings in $\mathcal{I}$.

A sufficient condition for the existence of the ranges defined in (8.6) on the basis of the intersection operation is the *convexity* of the sets examined above (see also Proposition 16).

Proposition 18 *For every set $\mathcal{I}$ of p-conditionings $\mathcal{M}_{\mathcal{T},\mathcal{I}}$ is a convex set.*

8.6 Probabilistic Constraints

We are mainly interested in *probabilistic constraints* which correspond to the formal model introduced above, which are *locally* defined and therefore *context-related*, and which *derive* and *refine* p-conditionings and check in this way the consistency of the whole knowledge base. In the framework of this paper we restrict ourselves to two simple *triangular cases* that take into account three concept expressions and restrict the extensional relations—and consequently the p-conditionings and the class of possible probabilities, using the terminology of the above section—still possible between these concepts.

Proposition 19 *Assume concepts A, B, C, and p-conditionings*

$$\mathcal{I} = \{A \stackrel{[p_l,p_u]}{\rightarrow} C, A \stackrel{[q_l,q_u]}{\rightarrow} B, B \stackrel{[q_l',q_u']}{\rightarrow} A, C \stackrel{[p_l',p_u']}{\rightarrow} A, p_l' = 0, q_l \neq 0\}.$$

The minimal p-conditioning $B \stackrel{R_{\min}}{\rightarrow} C \in \min_{\mathcal{T}}(\mathcal{I})$ derivable on the basis of this knowledge has the minimal range

$$R_{\min} = \left[\frac{q_l'}{q_l} \cdot \max(0, q_l + p_l - 1), \min\left(1,\ 1 - q_l' + p_u \cdot \frac{q_l'}{q_l}\right)\right]. \tag{8.9}$$

Note that since above proposition is based on the assumption $p_l' = 0, q_l \neq 0$ we already consider a special case; the more general but also complex constraints can be found in [Heinsohn, 1993]. If we use (8.9) in the framework of our example for deriving new p-conditionings, for instance, the assertions "Most ($\geq 95\%$) birds move by flying. An antarctic bird is a bird (subsumption). 20% of the birds are antarctic birds" lead to the inference "Several ($\geq 75\%$) antarctic birds move by flying".

The following proposition shows that in the case of logically interrelated concepts (below because of concept negation and conjunction) probabilistic constraints have to be further strengthened:

Proposition 20 *Assume concepts $A, B, C \in \mathcal{C} \setminus \{\bot\}$. Then the following relations hold:*

$$\left(B \stackrel{[p_l,p_u]}{\rightarrow} \neg B\right) \in \min_{\mathcal{T}}(\mathcal{I}) \quad \Rightarrow \quad p_u = 0$$

$$\left(A \stackrel{[p_l,p_u]}{\rightarrow} B\right) \in \min_{\mathcal{T}}(\mathcal{I}) \quad \Leftrightarrow \quad \left(A \stackrel{[1-p_u,1-p_l]}{\rightarrow} \neg B\right) \in \min_{\mathcal{T}}(\mathcal{I})$$

$$\left(A \stackrel{[p_l,p_u]}{\rightarrow} C\right) \in \min_{\mathcal{T}}(\mathcal{I}) \quad \Leftrightarrow \quad \left(A \stackrel{[p_l,p_u]}{\rightarrow} A \sqcap C\right) \in \min_{\mathcal{T}}(\mathcal{I})$$

The main advantage of examining local *triangular cases* is that "most" of the inconsistencies are discovered early and can be taken into account just in the *current context* of the three concepts involved. Further, not as yet known p-conditionings can be generated and the associated probability ranges can be stepwise refined. In the general case, testing probabilistic consistency leads for every p-conditioning to successively computing the intersections of the probability ranges derived on the basis of different local examinations.

By explicitly introducing restrictions for the ranges derived by instantiating the consistency requirements, *exceptions* can also be handled. For illustration, consider the above example situation in which the p-conditioning

$$antarctic\_bird \overset{[0.75,1]}{\rightarrow} flying\_object$$

is derived. In the absence of further information, all that can be concluded for the "flying proportion of *penguin*s" is the range $[0, 1]$. If a derived range is considered not to fit the subconcept, the range can be restricted further. For example, "no penguins fly" is represented by the p-conditioning $penguin \overset{0}{\rightarrow} flying\_object$, which satisfies the p-conditioning, $p \in [0, 1]$, obtained from consistency tests. In the categorical cases this corresponds to the overriding of properties in nonmonotonic inheritance networks.

8.7 Related Work

The importance of providing an integration of both term classification and uncertainty representation[2] was recently emphasized in some publications. However, they differ from each other and also from our proposal. Yen and Bonissone [Yen and Bonissone, 1990] consider this integration from a general point of view which, for instance, does not require a concrete uncertainty model,[3] while in our approach specific properties of an integration are demonstrated, based on a concrete probabilistic model. In [Yen, 1991] Yen proposes an extension of term subsumption languages to fuzzy logic that aims at representing and handling vague concepts. His approach generalizes a subsumption test algorithm for dealing with the notion of vagueness and imprecision. Since the language $\mathcal{ALCP}$ aims at modeling uncertainty, it already differs from Yen's proposal in its general objectives. Saffiotti [Saffiotti, 1990] presents a hybrid framework for representing epistemic uncertainty. His extension allows one to model uncertainty about categorical knowledge, e.g., to express one's belief on quantified statements such as "I am fairly (80%) sure that all birds fly". Note the difference from "I am sure that 80% of birds fly", which is modeled in this paper and requires a completely different formal basis. The work of Bacchus [Bacchus, 1990] is important because he not only explores the question of how far one can go using *statistical* knowledge but also presents LP, a logical formalism for representing and reasoning with statistical knowledge. In spite of being closely related to our work and being able to represent conditional probabilities, Bacchus does not provide a deep discussion of conditionals and the associated local consistency requirements.

---

[2] Brachman [Brachman, 1990] considers "probability and statistics" as one of the "potential highlights" in knowledge representation.

[3] A detailed overview on uncertainty and vagueness, and analyses are given in [Kruse *et al.*, 1991].

8.8 Conclusions

We have proposed the language $\mathcal{ALCP}$, a probabilistic extension of terminological logics that takes into account uncertain knowledge arising when certain concept properties are, e.g., usually but not categorically true. For this purpose, the notion of *probabilistic conditioning* based on a statistical interpretation has been introduced. $\mathcal{ALCP}$ allows the modeling and representation of important fragments of general commonsense knowledge as well as knowledge stemming from concrete applications. Inference mechanisms have been formulated for these kinds of knowledge. The knowledge that can be handled in this way includes *terminological knowledge* covering term descriptions and *uncertain knowledge* that can be given in the form of general assertions about (not generally true) concept properties. In this way also anomalies and categorical exceptions can be taken into account. The newly developed formal framework for handling uncertainty is based on *classes of probabilities* that offer a modeling of *ignorance* as one special feature. *Probabilistic constraints* allow the context-related generation and refinement of p-conditionings and check the consistency of the knowledge base. More details about the language $\mathcal{ALCP}$ concerning the uncertainty model for concepts presented in this paper and the associated extension for *assertional knowledge* are given in [Heinsohn, 1993].

8.9 References

[Baader and Hollunder, 1991] Franz Baader and Bernhard Hollunder. KRIS: knowledge representation and inference system. *SIGART Bulletin*, 2(3):8–14, 1991.

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

[Borgida *et al.*, 1989] Alexander 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 Mangement of Data*, pages 59–67, Portland, Oreg., June 1989.

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

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

[Brachman, 1990] Ronald 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, Mass., 1990.

[Heinsohn *et al.*, 1993] Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 1993. To appear. A preliminary version is available as DFKI Research Report RR-92-16.

[Heinsohn, 1991] Jochen Heinsohn. A hybrid approach for modeling uncertainty in terminological logics. In R. Kruse and P. Siegel, editors, *Symbolic and Quantitative Approaches to Uncertainty, Proceedings of the European Conference ECSQAU*, Lecture Notes in Computer Science 548, pages 198–205. Springer, Berlin, Germany, 1991.

[Heinsohn, 1992] Jochen Heinsohn. ALCP - an integrated framework to terminological and noncategorical knowledge. In *Proceedings of the 4th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'92)*, pages 493–496, Palma de Mallorca, Spain, July 6–10 1992.

[Heinsohn, 1993] Jochen Heinsohn. *ALCP - Ein hybrider Ansatz zur Modellierung von Unsicherheit in terminologischen Logiken.* Dissertation, Universität des Saarlandes, 1993.

[Kruse *et al.*, 1991] R. Kruse, E. Schwecke, and J. Heinsohn. *Uncertainty and Vagueness in Knowledge Based Systems: Numerical Methods.* Series Artificial Intelligence. Springer, Berlin, Germany, 1991. (491 pages).

[MacGregor and Bates, 1987] Robert MacGregor and Raymond Bates. The Loom knowledge representation language. Technical Report ISI/RS-87-188, University of Southern California, Information Science Institute, Marina del Rey, Cal., 1987.

[Nebel, 1990] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Computer Science, Volume 422. Springer, Berlin, Germany, 1990.

[Nebel, 1991] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In J. Sowa, editor, *Principles of Semantic Networks*, pages 331–362. Morgan Kaufmann, San Mateo, Cal., 1991.

[Owsnicki-Klewe, 1988] Bernd Owsnicki-Klewe. Configuration as a consistency maintenance task. In W. Hoeppner, editor, *Proceedings of the 12th German Workshop on Artificial Intelligence (GWAI-88)*, pages 77–87. Springer, Berlin, Germany, 1988.

[Patel-Schneider *et al.*, 1990] Peter F. Patel-Schneider, Bernd Owsnicki-Klewe, Alfred Kobsa, Nocola Guarino, Robert MacGregor, William S. Mark, Deborah McGuinness, Bernhard Nebel, Albrecht Schmiedel, and John Yen. Term subsumption languages in knowledge representation. *The AI Magazine*, 11(2):16–23, 1990.

[Patil *et al.*, 1992] Ramesh S. Patil, Richard E. Fikes, Peter F. Patel-Schneider, Don McKay, Tim Finin, Thomas Gruber, and Robert Neches. The DARPA knowledge sharing effort: Progress report. In B. Nebel, W. Swartout, and C. Rich, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference*, pages 777–788, Cambridge, MA, October 1992. Morgan Kaufmann.

[Saffiotti, 1990] 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, Mass., 1990.

[Schmidt-Schauß and Smolka, 1991] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1), 1991.

[Sigart Bulletin, 1991] *SIGART Bulletin: Special Issue on Implemented Knowledge Representation and Reasoning Systems*, volume 2(3). ACM Press, June 1991.

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

[Yen, 1991] J. Yen. Generalizing term subsumption languages to fuzzy logic. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

9

The Rule-Based Multi-Agent System MAGSY

Klaus Fischer[1]

ABSTRACT

This paper presents the rule-based multi-agent system MAGSY. The kernel of an agent in MAGSY is a forward-chaining rule interpreter. Each agent in MAGSY is a self-contained entity with a unique identification. If an agent knows the identification of another agent, it is able to send this agent messages. In doing so the agents perform services and call on services of other agents. Agents are able to create new agents dynamically. If an agent creates a new agent, it automatically knows the identification of this new agent. The knowledge of the agents is structured in an object-oriented knowledge representation scheme. There is a global knowledge base which contains the knowledge that may be accessed by all of the agents. Agents may store their identification in this global knowledge base and thus become known to all agents in the system. The paper describes the theoretical concepts of MAGSY as well as the language which was specially designed to specify multi-agent systems.

9.1 Introduction

The rule-based multi-agent system MAGSY was developed in subproject A2 *Task-Oriented Programming Techniques* of research project 331 *Information Processing in Autonomous Mobile Robot Systems* at the Technische Universität in Munich. The definition of the term *agent* in MAGSY meets the definition given by [Martial 92]. An agent is viewed as an abstract (computational) object. The kernel of each MAGSY agent is a forward-chaining rule interpreter. Each agent has therefore the problem solving capacity of an expert system.

In research project 331 problems are studied which are likely to occur in a factory of the future where autonomously acting systems (robots, machines, flexible transport systems etc.) solve daily tasks. In order to model these autonomously acting systems it is very important that they are able to exchange knowledge and to react to the actions of each other. This means for the planning components of such a system that they must be able to react to external events which occur asynchronously at any point in time. We call the ability to receive knowledge at any point in time and to react immediately to this knowledge *asynchronous knowledge access*.

MAGSY was first developed for the implementation of the task-oriented programming interface of such an autonomously acting system in a flexible manufacturing system. MAGSY agents have the capability of asynchronous knowledge access when they are using the communication primitives built into the rule-based language. MAGSY was implemented using C++ in a UNIX environment. The communication layer uses the stream port/socket interface of TCP/IP. Therefore, the agents of an application can be distributed on any number of different computers in a local area network.

Although MAGSY was designed for modelling autonomous systems in a flexible manufac-

[1]Project AKA-MOD, DFKI Saarbrücken, email: kuf@dfki.uni-sb.de

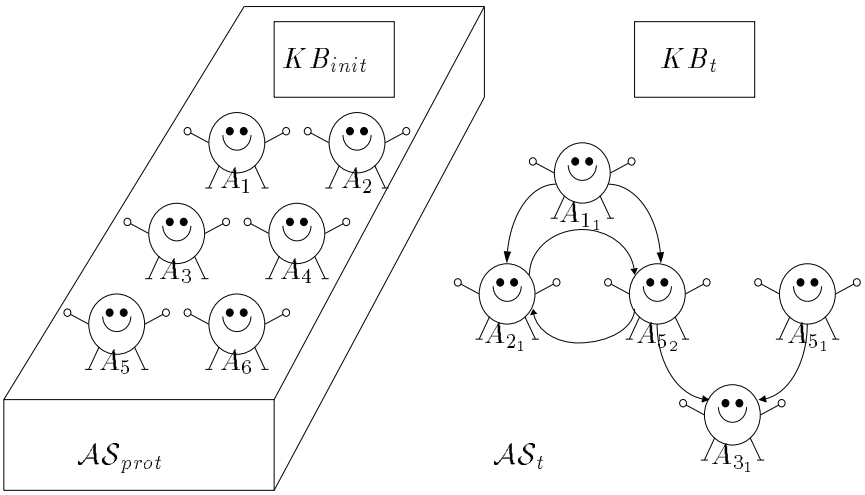

FIGURE 9.1. Static specification $\mathcal{AS}_{prot}$ of the solution for a problem and dynamic execution $\mathcal{AS}_t$.

turing system, it is a universal development tool for multi-agent applications. In addition to the application at the Technische Universität in Munich, MAGSY is used in research project AKA-MOD at the DFKI in Saarbrücken to implement two multi-agent applications. In the first application a multi-agent system for the transportation domain is implemented. Several transportation companies each of them controlling a set of trucks execute client orders. Each transportation company and each truck is represented by a MAGSY agent. In order to find good solutions to the problem the transportation companies have to cooperate closely with each other and with their trucks. In the second application a multi-agent system simulating a loading dock is developed. A set of forklifts have to load or unload a truck. The forklifts act autonomously in this environment. They decide by themselves which task to do next and how to execute it. In doing so they have to react to the actions of each other. All forklifts and the truck are represented by MAGSY agents. With the help of MAGSY it was possible for both applications to quickly implement a first prototype.

9.2 The Definition of the Multi-Agent System

This section describes the theoretical concept MAGSY is based on. In MAGSY an agent is defined by a triple $(\mathcal{F}, \mathcal{R}, \mathcal{T})$, where

- $\mathcal{F}$ is a set of facts which represent the local knowledge of the agent.
- $\mathcal{R}$ is a set of rules which define the strategies for the general behaviour of the agent.
- $\mathcal{T}$ is a set of services which are provided by the agent.

For a given agent A, $A_\mathcal{F}, A_\mathcal{R}$ and $A_\mathcal{T}$ denote the respective sets. Agent A may receive messages which change the set of facts or activate a service of the agent. Each service of an agent consists of a set of rules which become active when the specific service has been activated by an incoming message. The execution of a service by an agent A may change the set of facts $A_\mathcal{F}$ and the set of rules $A_\mathcal{R}$. Therefore, the agents are learning when they execute their services, which can also mean that rules and facts can be forgotten, i.e. are removed. It is also possible that the execution of a service by agent A creates a number of new agents. The requests for services which are received by an agent are treated according to

a specific strategy. Only one service can be active at a time. The other requests are queued and activated when the current service is finished. A priority is assigned to each service. A service s may be interrupted by a service i if a message arrives which activates i and i has a higher priority than s. If i is finished, s is reactivated. This process may happen recursively. At any point in time is therefore the service with highest priority active. If more than one request for this priority class is present, the request belonging to the oldest message (that is the earliest one received) will be executed.

The knowledge of the agents is represented in an object-oriented knowledge representation scheme. Objects are the basic elements which build a knowledge base. Objects which have the same structure are grouped together and make up classes. The description of a class is a prototype object which specifies how the elements (we call these elements instances or facts) look like. The properties of these instances are described by attributes. The classes are structured in a hierarchy in which attributes are inherited from the upper, more general, classes to the lower, more specific ones.

To model the solution of a given problem, a set of cooperating agents is specified. The static description of the solution of the problem is given by the prototype agent system $\mathcal{AS}_{prot} := (\mathcal{A}_{prot}, KB_{init})$, where

$\mathcal{A}_{prot}$	is the set of prototypical agents, the agents which may arise dynamically in the system. These agents are the potentially available problem solvers. Several instances of a specific prototypical agent can be created.
KB_{init}	is the initial state of the global knowledge base.

The process of problem solving starts with the initial agent system

$$\mathcal{AS}_{init} = ((A_{1_1}, \ldots, A_{n_1}), KB_{init}),$$

where $A_{1_1}, \ldots, A_{n_1} \in \mathcal{A}_{prot}$. The computation goes on while the agents perform requested services and call on services of other agents. In doing so the agents are cooperating by sending messages to each other. New agents may be created and killed again later on. A message may contain the identification of an agent, thus any desired structure of communication can be created dynamically. The contents of the global knowledge base can be accessed by all agents. Thus, the global knowledge base may be changed by the agents in any desired manner. For example, an agent can make its own identification known to all of the other agents by writing it into the global knowledge base.

In the example of figure 9.1 the system started with $\mathcal{AS}_{init} = ((A_{1_1}, A_{5_1}), KB_{init})$. A_{1_1} creates A_{2_1} and A_{5_2} and A_{5_1} creates A_{3_1}. At time t, A_{1_1} told A_{5_2} the identification of A_{2_1} and A_{5_2} told A_{2_1} its own identification. The identification of A_{3_1} came known to A_{5_2} because A_{3_1} stored its identification in the global knowledge base and A_{5_2} extracted this information from the global knowledge base. In this example there is no other way for A_{5_2} to get the identification of A_{3_1}.

A specification of a multi-agent system consists of the descriptions of the initial state of the global knowledge base and of a set of prototypical agents. The syntax of the description of the global knowledge base in Extended Backus-Naur Form (EBNF) is given by

GLOBAL_KNOWLEDGE ::= 'begin' 'global' 'knowledge'
{ CLASS_DESCRIPTION }
'end' 'global' 'knowledge'.

The syntax of an agent description is given by

```
AGENT_DESCRIPTION ::= 'agent' IDENTIFIER
                      { CLASS_DESCRIPTION }
                      'general' 'behaviour' { RULE }
                      SERVICES
                      INITIALIZATION.
SERVICES ::= { service IDENTIFIER { IDENTIFIER ':' VARIABLE }
               { RULE }
               'end' }.
INITIALIZATION ::= [ 'initialize' { INIT_ACTION } ] 'end'.
INIT_AKTION ::= A 'make' or a 'create-agent' action as it is
                specified in section 9.4.2.
VARIABLE ::= '<' IDENTIFIER '>'.
```

The concrete syntax of a class description and a rule will be given in section 9.3 and 9.4. IDENTIFIER is an alphanumeric character string beginning with an alpha character. All variables specified after the service name of a service are bound when the service is activated. They may be used as global bound variables in the rules of a service (see also section 9.4).

9.3 Knowledge Representation

The knowledge representation scheme used in MAGSY is similar to the one presented in [Reimer 89]. The set of all knowledge bases is given by:

$$\mathcal{W} =_{\text{def}} \{f \mid f : \mathcal{N} \rightarrow \mathcal{C} \cup \{\bot\}\}$$

where $\mathcal{N}$ is the set of all names and $\mathcal{C}$ the set of all class descriptions. A concrete knowledge base $w \in \mathcal{W}$ assigns to each name $n \in \mathcal{N}$ a class description or the value $\bot$. $w.n$ may be used as a shorthand for $w(n), n \in \mathcal{N}$. $\bot$ is defined by:

$$\bot : \mathcal{N} \rightarrow \{\bot\}.$$

The set of all class descriptions is given by:

$$\mathcal{C} =_{\text{def}} \{(super, attr, inst) \mid super \in \mathcal{N}, attr : \mathcal{N} \rightarrow \mathcal{V} \cup \{\bot\}, inst : \mathcal{N} \rightarrow \mathcal{I} \cup \{\bot\}\}$$

The components of a class description are selected by the functions *super*, *attr*, *inst*. Again $w.n.inst$ may be used as a shorthand for $inst(w(n)), n \in \mathcal{N}$. The same is to be allowed for the selector functions *super* and *attr*. For a concrete knowledge base w the following holds:

$$\begin{aligned} \forall n \in \mathcal{N} : w.n \neq \bot \quad &\Rightarrow_{\text{def}} \quad w.n.super \neq \bot \\ \forall w \in \mathcal{W}, \forall cn, \widehat{cn} \in \mathcal{N} : \widehat{cn} = w.cn.super \quad &\Rightarrow_{\text{def}} \quad w.\widehat{cn} \neq \bot \end{aligned}$$

The set of all instances of a knowledge base is given by:

$$\mathcal{I} =_{\text{def}} \{f \mid f : \mathcal{N} \mapsto \mathcal{V} \cup \{\bot\}\}$$

The set of all values $\mathcal{V}$ is defined by:

$$\begin{aligned}\mathcal{V} \;=_{\text{def}}\; & \textbf{char} \cup \textbf{string} \cup \textbf{int} \cup \textbf{real} \cup \textbf{bool} \cup \mathcal{N} \cup \\ & \textbf{listof(char)} \cup \textbf{listof(string)} \cup \textbf{listof}(\mathcal{N}) \cup \\ & \textbf{listof(int)} \cup \textbf{listof(real)} \cup \\ & \textbf{enumof(char)} \cup \textbf{enumof(string)} \cup \textbf{enumof}(\mathcal{N}) \cup \\ & \textbf{enumof(int)} \cup \textbf{enumof(real)}\end{aligned}$$

The set **string**[2] has to be distinguished from the set $\mathcal{N}$. $n \in \mathcal{N}$ can be converted to an element of **string** with the help of function *str*. In the same way an element $s \in$ **string** may be converted into an element of $\mathcal{N}$ by the function *name*. To give an intuitive understanding of these formal definitions, all the functions defined may be represented by tables which assign concrete values to a set of names. If a name is not listed in the table, the value $\perp$ is assigned to this name by the corresponding function.

There are some restrictions to the already given definitions. First, a useful relation is specified:

$$\begin{aligned}&\forall w \in \mathcal{W}, \forall x, y \in \mathcal{N} : is\_superclass(w.x, w.y) \Leftrightarrow_{\text{def}} \\ &\quad x = w.y.super \vee \exists z \in \mathcal{N} : z = w.y.super \wedge \; is\_superclass(w.x, w.z)\end{aligned}$$

There is one predefined class which has the name $object \in \mathcal{N}$. For this class the following holds

$$\begin{aligned}w.object.super \;&=_{\text{def}}\; object \\ \forall w \in W, \forall x \in \mathcal{N} : x \neq object \;&\Rightarrow_{\text{def}}\; \neg is\_superclass(w.x, w.x)\end{aligned}$$

There are two important definitions for instances of classes.

1. An instance of a given class x assigns to each name a value which is specified as an attribute name in the class description.

$$\forall w \in \mathcal{W}, \forall x, i, a \in \mathcal{N} : w.x.attr.a \neq \perp \wedge \; w.x.inst.i \neq \perp \Rightarrow_{\text{def}} w.x.inst.i.a \neq \perp$$

2. An instance of a given class x assigns to each name a value which is specified as an attribute name in the class description of a super class of x.

$$\begin{aligned}&\forall w \in \mathcal{W}, \forall x, y, i, a \in \mathcal{N} : \\ &\quad is\_superclass(w.x, w.y) \wedge \; w.x.attr.a \neq \perp \wedge w.y.inst.i \neq \perp \\ &\quad \Rightarrow_{\text{def}} w.y.inst.i.a \neq \perp\end{aligned}$$

   In the description of a class only one super class can be specified, normally this is called single inheritance. If it would be possible to specify more than one super class in the class description, one would speak of multiple inheritance.

[2] Words written in bold font are syntactical keywords of the language in which the agents of MAGSY are specified

Till now the abstract semantics of the knowledge representation scheme used in MAGSY was described. A concrete class description is specified following the syntax:

CLASS_DESCRIPTION ::= 'class' IDENTIFIER
['super' 'class' IDENTIFIER]
'attributes' { IDENTIFIER ':' TYPE ';' }
['instances' { INSTANCE }]
'end'
TYPE ::= BASIC_TYPE | 'enumof' '(' { VALUE } ')' | 'listof' '(' { VALUE } ')'
BASIC_TYPE ::= 'char' | 'string' | 'int' | 'real' | 'bool'
INSTANCE ::= '(' IDENTIFIER IDENTIFIER { IDENTIFIER ':' VALUE } ')'
VALUE specifies an object $\in \mathcal{V}$.

9.4 Definition of the Rule Interpreter

The kernel of an agent in MAGSY is a forward-chaining rule interpreter. A rule consists of a condition and an action part. If the condition part of a rule is fulfilled, the rule may fire, that means the actions in the action part may be executed. The actions in the action part of a rule are not logical consequences derived from the condition part of the rule. They are just actions which were specified by a system engineer and which can be executed if the condition part of the rule is fulfilled. A rule is constructed according to the following scheme:

(**rule** rule-name (priority)
E($< x_1 >, \ldots, < x_n >, < f_1 >, \ldots, < f_m >$)
N($< x_1 >, \ldots, < x_n >$)
test-attributes(P($< x_1 >, \ldots, < x_n >$))
$\longrightarrow$
A($< x_1 >, \ldots, < x_n >, < f_1 >, \ldots, < f_m >$)
)

where $m, n \in I\!N$. 'rule-name' is an identifier which may be chosen freely. 'priority' is a number (positive and negative values are allowed) which determines the priority of the rule. The priority of the rules is taken into account if more than one rule may be executed in an agent. The normal ordering $<$ on natural numbers is used, this means the priority is higher and therefore the rule is more likely to be chosen if the number is bigger. If the priority is omitted, the rule gets the default value 0 as priority. It is therefore possible to give a higher or a lower priority to the rule with respect to the normal case. E($< x_1 >, \ldots, < x_n >$) and N($< x_1 >, \ldots, < x_n >$) represent a sequence of positive or negative conditions respectively (existential E or non-existential N conditions). P($< x_1 >, \ldots, < x_n >$) is a quantifier-free expression in first order logic which may contain variables $< x_1 >, \ldots, < x_n >$ which have been bound in the positive conditions of the rule. The logical expression P(...) must be fulfilled in addition to the conditions of the rules and puts therefore additional constraints to the values of the variables $< x_1 >, \ldots, < x_n >$.

The positive conditions are checked in the order in which they are written down and have the following general form:

isfact(class instance attribute$_1$: value$_1$, ..., attribute$_m$: value$_m$):<*f*>

where $m \in I\!N$. 'class' represents the class name, 'instance' the instance name and 'attribute$_i$' represents the attribute name with respect to the knowledge representation scheme presented in section 9.3. For 'value$_i$' a constant value, a function call or a variable may be specified. The first time they occur the variables $< x_i >$ are bound to the respective attribute value of the fact which fulfils the condition. If the variable <*f*> is specified, it refers to the fact which fulfils the condition. A concrete example of a positive condition is given by:

isfact(*agent* <*i*> *type*: active)

where <*i*> is bound to the instance name of the fact that fulfils the condition. The formal semantics of this condition is:

$$\exists i \in \mathcal{N} : w.agent.inst.i.type = active$$

A rule may fire only if there is no fact which matches one of its negative conditions. Negative conditions have the following general structure:

not isfact(class instance attribute$_1$: value$_1$ ... attribute$_m$: value$_m$)
with(P($< y_1 >$, ..., $< y_k >$, $< x_1 >$, ..., $< x_n >$))

where $k, m, n \in I\!N$. 'class' represents the class name, 'instance' represents the instance name and 'attribute$_i$' represents the attribute name with respect to the knowledge representation scheme presented in section 9.3. Instead of 'value$_i$' a constant, a variable $< y_i >$ or a variable $< x_i >$ may be specified, where $< x_i >$ must have been already bound by a positive condition. P($< y_1 >$, ..., $< y_k >$, $< x_1 >$, ..., $< x_n >$) is a quantifier-free expression in first order logic. It adds additional constraints to the facts which are able to fulfil the **isfact** clause of this negative condition and thus make the whole condition evaluate to **false**. All variables $< y_1 >$, ..., $< y_k >$ must have been bound in the **isfact** clause of the negative condition. A concrete example for a negative condition is:

not isfact(*candidate* <*i*> *difference*: <*x*>) **with**(<*x*> < 5)

where the attribute difference of class candidate is of type **int** and const specifies a constant number which might be given by an already bound variable (see rule choice in the example in the appendix). This condition has the semantics:

$$\forall i \in \mathcal{N} : w.candidate.inst.i.difference \geq 5$$

To determine which rule is able to fire and for selecting one rule if more than one rule is able to fire the well-known RETE algorithm is used [Forgy 82].

9.4.1 The Representation of Facts

Facts are represented by instances of classes in the knowledge representation scheme presented in section 9.3. There is a distinction between global and local knowledge or, to put it the other way round, between global and local facts. There is a special instance name **self** $\in \mathcal{N}$. All facts with instance name **self** are purely local knowledge and are only known by the agent which creates such a fact using the **make** action (see below). On the other hand, all facts with an instance name $i \neq$ **self** specify global knowledge. This does not mean, that all of these facts are physically shared by all of the agents in the system. Facts with instance names $i \neq$ **self** are local copies of the instances in the global knowledge base. The precise

semantics of the creation and modification of global and local knowledge is explained with the semantics of the **make** and the **modify** action.

The conditions of rules can refer to facts with instance name **self**, with a constant instance name $i \neq$ **self** or with a variable instance name. If the instance name is a variable the agent must know all instances of the mentioned class. If the instance name is a constant $i \neq$ **self**, the agent only has to know this specific instance of the mentioned class. Almost the same is true for the class names, here a constant name or a variable may be specified. Therefore, an agent which contains a rule with a condition in which both the class and the instance name are specified by a variable will have to know the whole global knowledge, i.e. every instance which is created in the global knowledge base and every change which is performed during the system execution. During compilation time the rules of an agent are analysed by the compiler. Information produced by the compiler makes it possible for the rule interpreter to tell the global knowledge base, which instances of which classes a specific agent has to know about. The same is true in an analogous way for rules created at runtime. After the initialization of an agent the global knowledge base will send to this agent all currently present instances which match the specifications of this agent's demands. Furthermore, the global knowledge base will send all the changes with respect to these instances to the agent. This way of updating global knowledge guarantees that if all of the agents reach a point, where no more actions are performed, then, after all activities in the system have stopped, all local copies of global knowledge will be equal to the instances valid in the global knowledge base.

As in the case of a shared memory in a multi-processor computer, it is clear that updates in the global knowledge may lead to inconsistent states of the global knowledge base if only the concepts presented till now are used. If one wants to use the global knowledge base for synchronisation tasks, further concepts like lock mechanisms have to be introduced [Fischer 92b].

9.4.2 Actions Which May be Used

In this section some of the actions which might be used on the right-hand side of a rule are described. A full list of this actions can be found in [Fischer 92b]. First some more syntactical elements must be defined.

FACT_SPEC ::= IDENTIFIER IDENTIFIER ATTR_VAL_LIST
ATTR_VAL_LIST ::= IDENTIFIER ':' VALUE_SPEC
VALUE_SPEC ::= VALUE | IDENTIFIER (VALUE_SPEC) | VARIABLE

'make' '(' FACT_SPEC ')'

Section 9.4.1 already explained that the semantics of the **make**-action of the rule interpreter used in MAGSY differs from the one used by commonly known rule interpreters [Allen 83; Brownston 85]. When a fact with instance name **self** is created, this knowledge is purely local to the agent and the fact is inserted into the set of facts for this specific agent only. In the knowledge representation scheme, which was presented in section 9.3, each instance of a class needs a unique instance name. Therefore, when a fact is created with the instance name **self**, a unique instance name, which is unknown to the user, is created internally. The user is not able to create such an automatically produced instance name explicitly. If the class name is defined local to the agent, it is only possible to

create instances with instance name **self** for this classes.

If in FACT_SPEC an instance name $i \neq$ **self** is specified, the fact to be created is a global fact. In this case first of all the agent's local set of facts is checked if there is already an instance i in the class c specified by FACT_SPEC. If such an instance is present this instance is removed from the set of local facts. After that the newly created instance is sent to the global knowledge base. The global knowledge base then sends this newly created instance to all of the agents which need this knowledge and this of course means that the instance is also sent back to the agent which originally created it. Only when this answer message from the global knowledge base is received, the instance is inserted as a fact into the local set of facts of the agent. The whole process is done in a fully asynchronous manner. The **make** action is finished when the message to the global knowledge base is sent. Therefore, there may be an arbitrary long time between the **make** action and the actual insertion of the fact into the local set of facts but the agent is able to continue its work unless it really needs the newly created global information.

'remove' '(' VARIABLE { VARIABLE } ')'

The variables specified in the **remove** action must have been bound to facts in the condition part of the rule. This facts are removed from the agent's local set of facts.

'modify' '(' VARIABLE ATTR_VAL_LIST ')'

The semantics of the **modify** action is defined by the **remove** and **make** action. The fact the specified variable refers to is removed from the agent's local set of facts using the **remove** action. After that a new fact is created using the **make** action. This new fact is computed using the old fact referred to by the specified variable and the ATTR_VAL_LIST specified in the **modify** action. Here the attribute value of the old fact is copied into the new one if this attribute is not specified in ATTR_VAL_LIST. In ATTR_VAL_LIST it is impossible to specify a class or an instance name and therefore it is impossible to change the class or the instance name of a fact using the **modify** action. All statements about local and global facts mentioned for the **make** action are also true for the **modify** action.

Now those actions are listed which were introduced to make it possible for the agents to communicate with each other.

'create-agent' '(' AGENT_NAME [NODE] ')'

A new agent is created which is executed in parallel to the agent which calls the action. A fact which specifies the identification of the new agent is inserted into the creator's local set of facts. AGENT_NAME must be a value specification, which evaluates to a symbol and specifies the name of the agent which will be created. By specifying NODE, which must be a value specification that evaluates to a symbol, the system engineer can determine on which node in the network the new agent will be created. Here the symbol LOCAL or REMOTE may also be specified. In the first case the agent is created on the node on which the creator is executed and in the second case it is created on a different node.

'request-service' '(' AGENT_ID SERVICE_NAME ATTR_VAL_LIST)

This action sends a request for the service specified by SERVICE_NAME, a value specification evaluating to a symbol, to the agent with identification AGENT_ID,

a value specification evaluating to a symbol which must specify a valid identification of an agent. The agent which calls this action is able to continue its work immediately without waiting for the agent specified by AGENT_ID to respond to the service request. ATTR_VAL_LIST specifies how the variables specified in the service declaration of the agent specified by AGENT_ID will be instantiated when the service is activated in this agent.

'send-fact' '(' AGENT_ID FACT_SPEC ')'
: This action inserts asynchronously the fact specified by FACT_SPEC into the local set of facts of the agent specified by AGENT_ID, a value specification evaluating to a symbol which must specify a valid identification of an agent. The communication interface guarantees, that the semantics of the fact that has been sent is not changed.

'send-answer' '(' FACT_SPEC ')'
: The semantics of this action is the same as the semantics of the **send-fact** action. This action may only be used in rules of services because the receiver of the fact is the client of the service and therefore in general unknown to the agent which executes the service.

9.5 Conclusion

This paper presented the rule-based multi-agent system MAGSY. MAGSY is a high-level development tool for the design of multi-agent applications. Compared with other multi-agent development tools already presented in literature, e.g. MACE [Gasser 88], MAGSY has the advantage that its agents can be excuted in a standard UNIX environment where they can be distributed on any number of computers in the internet network. MAGSY combines the advantages of frame-based knowledge representation [Minsky 75; Reimer 89], message passing communication facilities of actor systems [Agha 86], and the efficiency of production systems which are based on the RETE algorithm [Forgy 82].

The rule-based programming style used for the description of each agent is highly related to Petri nets and predicate/transition nets [Fischer 92a]. Each agent may be viewed as a predicate/transition net. Because the agents are able to communicate with each other they are able to influence each others marking by exchanging tokens (may be one should better say terms for predicate/transition nets). The relation between rule-based programs and Petri nets or predicate/transition nets are interesting because Petri nets and predicate/transition nets are theoretically well understood and for certain net types it is possible to do correctness proofs automatically [Starke 90]. [Tauber 88] showed on the other hand a high level equivalency of predicate/transition nets and other well known abstract programming languages for the description of concurrent processes like CCS [Milner 80] and TCSP [Brookes 84]. Therefore, we can state that there is implicit parallelism in each MAGSY agent due to the rule-based programming style, which is highly related to Petri net or predicate/transition net descriptions for concurrent execution. In MAGSY there is additionally explicit representation of parallelism because each agent is a self-contained entity represented by a process in the UNIX operating system.

There is additional a completely different view on the MAGSY system. It is well known that rule-based systems have internally the architecture of a blackboard system [Engelmore 88]. Because every agent in MAGSY is a self contained rule-based system, each MAGSY agent can be viewed as a blackboard system. In MAGSY there is additionally a global blackboard

which is represented by the global knowledge base. Therefore, each MAGSY application can be viewed as a distributed blackboard system, consisting of a global blackboard represented by the global knowledge base and a set of knowledge sources which are itself blackboard systems and which are represented by MAGSY agents.

9.6 References

[Agha 86] Gul A. Agha, *ACTORS: A Model of Concurrent Computation in Distributed Systems*, Series in Artificial Intelligence, The MIT Press, Cambridge, Massachusetts, 1986.

[Allen 83] Elisabeth M. Allen, "YAPS: Yet Another Production System", Technischer Bericht, University of Maryland, December 1983.

[Brookes 84] S.D. Brookes, C.A.R. Hoare und A.W. Roscoe, "A theory of communicating sequential processes", *Journal of the ACM*, 31:560–599, 1984.

[Brownston 85] L. Brownston, *Programming Expert Systems in OPS5*, Addison-Wesley, 1985.

[Engelmore 88] Robert Engelmore und Tony Morgan, Hrsg., *Blackboard Systems*, Addison-Wesley Publishing Company, 1988.

[Fischer 92a] Klaus Fischer, "Concepts for Hierarchical Planning in a Flexible Manunfacturing System", Technischer Bericht, DFKI — Workshop Planen —, 1992.

[Fischer 92b] Klaus Fischer, *Verteiltes und kooperatives Planen in einer flexiblen Fertigungsumgebung*, PhD thesis, TU München, Institut für Informatik, Juli 1992.

[Forgy 82] C.L. Forgy, "RETE — A Fast Algorithm for the Many Pattern – Many Object Pattern Match Problem", *Artificial Intelligence*, 19:17–37, 1982.

[Gasser 88] Les Gasser, Carl Braganza und Nava Herman, "Implementing Distributed AI Systems Using MACE", In Alan H. Bond und Les Gasser, Hrsg., *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, San Mateo, California, 1988.

[Martial 92] Frank v. Martial, "Einführung in die Verteilte Künstliche Intelligenz", *KI*, 1/92, 1992.

[Milner 80] R. Milner, *A calculus of communicating systems*, volume 92, Springer-Verlag, 1980.

[Minsky 75] Marvin Minsky, "A Framework for Representing Knowledge", In Patric H. Winston, Hrsg., *The Psychology of Computer Vision*, McGraw Hill, New York, 1975.

[Reimer 89] Ulrich Reimer, *FRM: Ein Frame-Repräsentationsmodell und seine formale Semantik*, volume 198 of *Informatik Fachberichte*, Springer-Verlag, 1989.

[Starke 90] Peter H. Starke, *Analyse von Petri-Netz-Modellen*, B.G. Teubner, Stuttgart, Juli 1990.

[Tauber 88] Dirk A. Tauber, *The finite representation of abstract programs by automata and Petri nets*, PhD thesis, TU München, Institut für Informatik, Dezember 1988.