An Empirical Analysis of Terminological Representation Systems*

Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
W-6600 Saarbrücken, Germany
e-mail: ⟨*last name*⟩@dfki.uni-sb.de

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.

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 operating 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

*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.

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.

The Experiment

The empirical analysis can be roughly divided into two parts (see Figure 1).[1] 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.

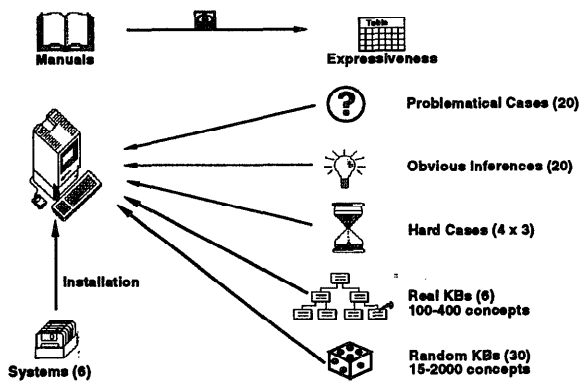

Figure 1: Experiment design

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 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.

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 systems we did not include (or mention) are not worthwhile to be analyzed. The only reason not to include all

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

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.

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*.[2]

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 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

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

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.

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 2.[3] They clearly indicate that the systems indeed exhibit a very rapid growth in the runtime.

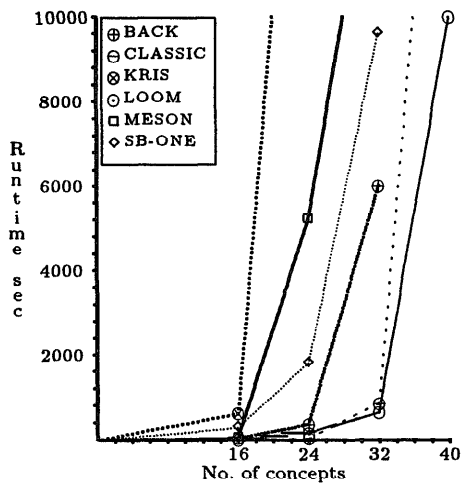

Figure 2: Runtime performance for hard cases

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 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

[3]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.

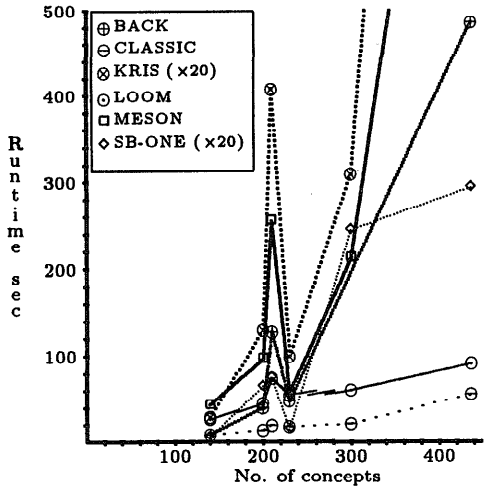

Figure 3: Runtime performance for realistic cases

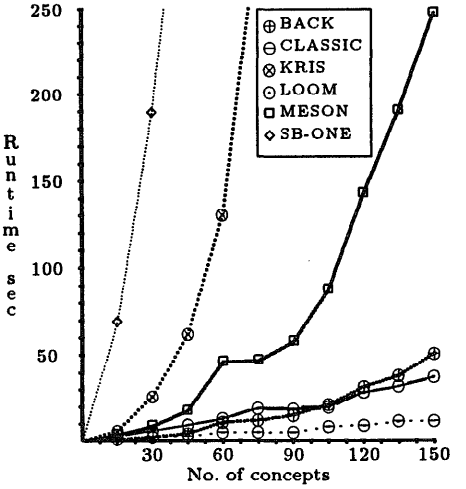

Figure 4: Runtime performance for small random KBs

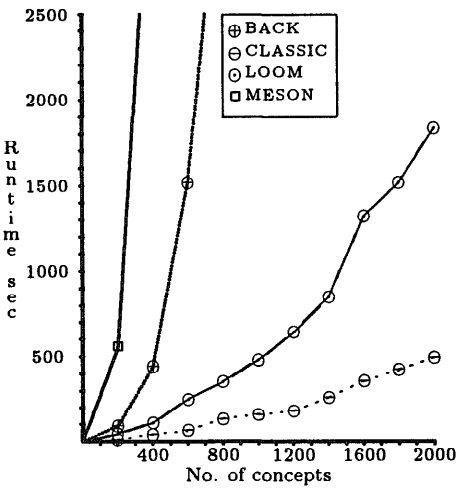

Figure 5: Runtime performance for large random KBs

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 3, 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.

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 3, 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 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, av-

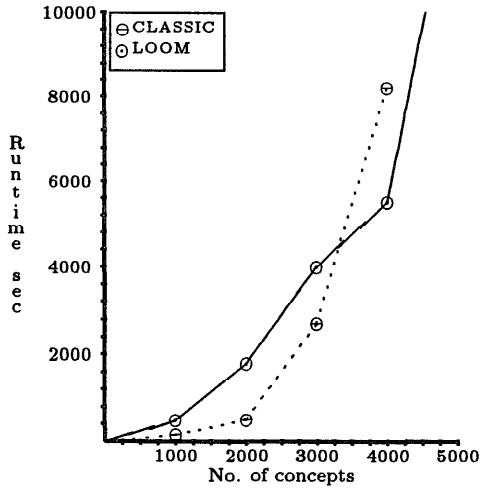

Figure 6: Runtime performance for very large random KBs

erage number of role restrictions, etc.). The results of this test are given in Figures 4, 5, and 6.

Comparing the curves in these three figures with the curves in Figure 3, 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.

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.

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.

References

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

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

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

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

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

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

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

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

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, 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.