Published in *Artificial Intelligence* 68(2): 367–397, 1994

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
D-66123 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 KL-ONE 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.

*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.

†This is a revised and extended version of a paper with the same title published in the Proceedings of the Tenth National Conference on Artificial Intelligence, San Jose, CA, 1992

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 [14] or as components of larger AI systems, such as natural language systems [37] or design systems [41]. Assuming that the application task is the configuration of computer systems [29], 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 drive* attached to it, for example. The main reasoning service provided by terminological representation systems is checking for *incoherent* 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 [11]. Meanwhile, the underlying framework has been adopted by various research groups, and more than a dozen terminological representation systems have been implemented [31]. 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 [39, 40]. 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

Figure 1: Experiment design

what performance to expect from existing terminological representation systems. These results complement the various analytical results on the computational complexity of terminological reasoning.

The rest of the paper is structured as follows. In the next section we give a sketch of the experiment design, and in Section 3 we briefly describe the systems we analyzed. In Section 4, we specify a "common terminological language" we used as a basis for comparing the expressiveness of the system. The qualitative results are given in Section 5, and the quantitative results are described in Section 6.

2 The Experiment

The empirical analysis can be roughly divided into two parts (see Figure 1). The first part covers qualitative facts concerning system features and expressiveness. In order to describe expressiveness, we first developed a "common terminological language" that covers a superset of all terminological languages employed in the systems we considered (see also [5]). The analysis of 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 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, runtime 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 runtime performance may have changed.

3 Systems

There are a large number of systems which could have been included in an empirical analysis, e.g., KL-ONE [11], LILOG [8], NIKL [36], K-REP [23], KRS [18], KRYPTON [10], YAK [12]. 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 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 [32] (Version 4.2, pre-released), CLASSIC [30] (Version 1.02, released), KRIS [6] (Version 1.0, experimental), LOOM [22] (Version of May 1990, pre-released), MESON [29] (Version 2.0, released), and SB-ONE [20] (Version of January 1990, released). Except as noted, all systems were written in COMMONLISP and tested on a MacIvory. Below we give a brief description of the systems:

- 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 [13]. 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 [14].

- KRIS has been developed by the WINO project at DFKI. In contrast to other systems, it provides different complete inference algorithms for very expressive languages. In our tests, we mainly used the algorithm for the language $\mathcal{ALCFN}$, except when noted otherwise. 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 [29]. 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.

4 The Common Terminological Language CTL

While all terminological representation systems are based on the same representational framework, they appear nevertheless to be quite different on the surface level. In order to be able to describe the expressiveness of the systems in a common framework and in order to translate knowledge bases from one formalism to another it was necessary to develop a "common terminological language" (CTL), which is specified below (see also [5]).

The common representational framework of terminological representation systems can be sketched as follows. There exists a number of symbols that belong to different syntactic categories. Usually, there are at least two disjoint categories, *concepts* and *roles*. Sometimes also *attributes* (also called *features*) and *individuals* are part of the vocabulary. Formally, we assume four disjoint alphabets, namely,

- the set of *concept symbols* $\mathbf{A}$, where A and B are used to denote elements of $\mathbf{A}$;
- the set of *role symbols* $\mathbf{P}$, where P and Q denote elements of $\mathbf{P}$;
- the set of *attribute symbols* (or *feature symbols*) $\mathbf{p}$, where p and q denote elements of $\mathbf{p}$.
- the set of *individual symbols* $\mathbf{I}$, where i and j denote elements of $\mathbf{I}$;

The intended meaning of a concept symbol is the extension of this concept. For instance, the concept symbol Workstation is intended to denote the set of all workstations. Similarly, role symbols denote two-place relations, attribute symbols denote functional relations, and individual symbols denote domain elements.

A number of *concept-* and *role-forming operators* can be used to form new concept and role expressions where the meaning of these expressions is determined compositionally from its parts (see Tables 1 and 2). Further, there are some operators to form so-called *terminological axioms* that are used to assign the meaning of an expression to a symbol, which are specified in Table 3. Finally, there may be some operators to specify additional *restrictions* on the interpretation of

Abstract form	Concrete form	Interpretation
$C, D \rightarrow A$	A	$[\![A]\!]^{\mathcal{I}}$
$\mid \top$	AnyThing	$\mathcal{D}$
$\mid \bot$	Nothing	$\emptyset$
$\mid C \sqcap D$	(and C D)	$[\![C]\!]^{\mathcal{I}} \cap [\![D]\!]^{\mathcal{I}}$
$\mid C \sqcup D$	(or C D)	$[\![C]\!]^{\mathcal{I}} \cup [\![D]\!]^{\mathcal{I}}$
$\mid \neg C$	(not C)	$\mathcal{D} \setminus [\![C]\!]^{\mathcal{I}}$
$\mid \forall R{:}C$	(all R C)	$\{d \in \mathcal{D} \mid [\![R]\!]^{\mathcal{I}}(d) \subseteq [\![C]\!]^{\mathcal{I}}\}$
$\mid \exists R$	(some R)	$\{d \in \mathcal{D} \mid [\![R]\!]^{\mathcal{I}}(d) \neq \emptyset\}$
$\mid \geq nR$	(atleast n R)	$\{d \in \mathcal{D} \mid \lvert[\![R]\!]^{\mathcal{I}}(d)\rvert \geq n\}$
$\mid \leq nR$	(atmost n R)	$\{d \in \mathcal{D} \mid \lvert[\![R]\!]^{\mathcal{I}}(d)\rvert \leq n\}$
$\mid nR$	(exact n R)	$\{d \in \mathcal{D} \mid \lvert[\![R]\!]^{\mathcal{I}}(d)\rvert = n\}$
$\mid (0 \vee \geq n)R$	(optatleast n R)	$\{d \in \mathcal{D} \mid [\![R]\!]^{\mathcal{I}}(d) = \emptyset \vee \lvert[\![R]\!]^{\mathcal{I}}(d)\rvert \geq n\}$
$\mid \exists R{:}C$	(some R C)	$\{d \in \mathcal{D} \mid [\![R]\!]^{\mathcal{I}}(d) \cap [\![C]\!]^{\mathcal{I}} \neq \emptyset\}$
$\mid \geq nR{:}C$	(atleast n R C)	$\{d \in \mathcal{D} \mid \lvert[\![R]\!]^{\mathcal{I}}(d) \cap [\![C]\!]^{\mathcal{I}}\rvert \geq n\}$
$\mid \leq nR{:}C$	(atmost n R C)	$\{d \in \mathcal{D} \mid \lvert[\![R]\!]^{\mathcal{I}}(d) \cap [\![C]\!]^{\mathcal{I}}\rvert \leq n\}$
$\mid nR{:}C$	(exact n R C)	$\{d \in \mathcal{D} \mid \lvert[\![R]\!]^{\mathcal{I}}(d) \cap [\![C]\!]^{\mathcal{I}}\rvert = n\}$
$\mid RC = SC$	(eq RC SC)	$\{d \in \mathcal{D} \mid [\![RC]\!]^{\mathcal{I}}(d) = [\![SC]\!]^{\mathcal{I}}(d)\}$
$\mid RC \neq SC$	(neq RC SC)	$\{d \in \mathcal{D} \mid [\![RC]\!]^{\mathcal{I}}(d) \neq [\![SC]\!]^{\mathcal{I}}(d)\}$
$\mid RC \subseteq SC$	(subset RC SC)	$\{d \in \mathcal{D} \mid [\![RC]\!]^{\mathcal{I}}(d) \subseteq [\![SC]\!]^{\mathcal{I}}(d)\}$
$\mid r : C$	(in r C)	$\{d \in \mathcal{D} \mid \emptyset \neq [\![r]\!]^{\mathcal{I}}(d) \subseteq [\![C]\!]^{\mathcal{I}}\}$
$\mid r : i$	(is r i)	$\{d \in \mathcal{D} \mid [\![i]\!]^{\mathcal{I}} \in [\![r]\!]^{\mathcal{I}}(d)\}$
$\mid rc = sc$	(eqopt rc sc)	$\{d \in \mathcal{D} \mid [\![rc]\!]^{\mathcal{I}}(d) = [\![sc]\!]^{\mathcal{I}}(d)\}$
$\mid rc \neq sc$	(neqopt rc sc)	$\{d \in \mathcal{D} \mid [\![rc]\!]^{\mathcal{I}}(d) \neq [\![sc]\!]^{\mathcal{I}}(d)\}$
$\mid rc \overset{\downarrow}{=} sc$	(eq rc sc)	$\{d \in \mathcal{D} \mid [\![rc]\!]^{\mathcal{I}}(d) = [\![sc]\!]^{\mathcal{I}}(d) \neq \emptyset\}$
$\mid rc \overset{\downarrow}{\neq} sc$	(neq rc sc)	$\{d \in \mathcal{D} \mid \emptyset \neq [\![rc]\!]^{\mathcal{I}}(d) \neq [\![sc]\!]^{\mathcal{I}}(d) \neq \emptyset\}$
$\mid \{i_1, i_2, \ldots, i_n\}$	(oneof $i_1 \ldots i_n$)	$\{[\![i_1]\!]^{\mathcal{I}}, [\![i_2]\!]^{\mathcal{I}}, \ldots, [\![i_n]\!]^{\mathcal{I}}\}$

Table 1: CTL: concept forming operators

symbols, for instance, that two concept symbols denote disjoint sets or that they cover the interpretation of another concept (see Table 4).

Additionally to the operators listed in the four tables, most systems also offer some "escape" mechanism to the host system in the form of *test* functions and *host data-types*. We have omitted these from the description of the languages because this escape mechanism does not belong to the core of the terminological languages.

In the first column of the tables, we specify the abstract syntax, which we use in the paper. In the second column we specify the concrete syntax that is similar to the actual syntax used in systems such as CLASSIC, KRIS, and LOOM. In the third column, the formal meaning of the operators is specified using a set-theoretic semantics. This kind of semantics is based on *interpretations* $\mathcal{I} = \langle \mathcal{D}, [\![\cdot]\!]^{\mathcal{I}} \rangle$, where

Abstract form	Concrete form	Interpretation
$R, S \rightarrow P$	P	$[\![P]\!]^{\mathcal{I}}$
$\mid \top$	AnyRelation	$\mathcal{D} \times \mathcal{D}$
$\mid R \sqcap S$	(and R S)	$[\![R]\!]^{\mathcal{I}} \cap [\![S]\!]^{\mathcal{I}}$
$\mid R \sqcup S$	(or R S)	$[\![R]\!]^{\mathcal{I}} \cup [\![S]\!]^{\mathcal{I}}$
$\mid R^{-1}$	(inverse R)	$\{(d, d') \mid (d', d) \in [\![R]\!]^{\mathcal{I}}\}$
$\mid r^{-1}$	(inverse r)	$\{(d, d') \mid (d', d) \in [\![r]\!]^{\mathcal{I}}\}$
$\mid R\vert_C$	(restr R C)	$\{(d, d') \in [\![R]\!]^{\mathcal{I}} \mid d' \in [\![C]\!]^{\mathcal{I}}\}$
$\mid {}_C\vert R$	(domrestr R C)	$\{(d, d') \in [\![R]\!]^{\mathcal{I}} \mid d \in [\![C]\!]^{\mathcal{I}}\}$
$\mid C \times D$	(domainrange C D)	$[\![C]\!]^{\mathcal{I}} \times [\![D]\!]^{\mathcal{I}}$
$\mid R^{+}$	(trans R)	$([\![R]\!]^{\mathcal{I}})^{+}$
$\mid RC$		
$\mid r$		
$RC \rightarrow (R_1, \ldots, R_n)$	(compose $R_1 \ldots R_n$)	$[\![R_1]\!]^{\mathcal{I}} \circ \ldots \circ [\![R_n]\!]^{\mathcal{I}}$
$\mid \mathbf{1}$	self	$\{(d, d) \mid d \in \mathcal{D}\}$
$r \rightarrow p$	p	$[\![p]\!]^{\mathcal{I}}$
$\mid r\vert_C$	(restr r C)	$\{(d, d') \in [\![r]\!]^{\mathcal{I}} \mid d' \in [\![C]\!]^{\mathcal{I}}\}$
$\mid rc$		
$rc \rightarrow (r_1, \ldots, r_n)$	(compose $r_1 \ldots r_n$)	$[\![r_1]\!]^{\mathcal{I}} \circ \ldots \circ [\![r_n]\!]^{\mathcal{I}}$
$\mid \mathbf{1}$	self	$\{(d, d) \mid d \in \mathcal{D}\}$

Table 2: CTL: role and attribute forming operators

Abstract form	Concrete form	Interpretation
$A \doteq C$	(defconcept A C)	$[\![A]\!]^{\mathcal{I}} = [\![C]\!]^{\mathcal{I}}$
$A \sqsubseteq C$	(defprimconcept A $\{C\}$)	$[\![A]\!]^{\mathcal{I}} \subseteq [\![C]\!]^{\mathcal{I}}$
$P \doteq R$	(defrole P R)	$[\![P]\!]^{\mathcal{I}} = [\![R]\!]^{\mathcal{I}}$
$P \sqsubseteq R$	(defprimrole P R)	$[\![P]\!]^{\mathcal{I}} \subseteq [\![R]\!]^{\mathcal{I}}$
$P \sqsubseteq \top$	(defprimrole P)	$[\![P]\!]^{\mathcal{I}} \subseteq \mathcal{D} \times \mathcal{D}$
$p \doteq r$	(defattribute p r)	$[\![p]\!]^{\mathcal{I}} = [\![r]\!]^{\mathcal{I}}$
$p \sqsubseteq \top$	(defprimattribute p)	$[\![p]\!]^{\mathcal{I}} \subseteq \mathcal{D} \times \mathcal{D}$

Table 3: CTL: terminological axioms

$\mathcal{D}$ is an arbitrary set and $[\![\cdot]\!]^{\mathcal{I}}$ is a function

$$[\![\cdot]\!]^{\mathcal{I}}\colon \begin{cases} \mathbf{I} & \rightarrow \mathcal{D} \\ \mathbf{A} & \rightarrow 2^{\mathcal{D}} \\ \mathbf{P} & \rightarrow 2^{\mathcal{D}\times\mathcal{D}} \\ \mathbf{p} & \rightarrow 2^{\mathcal{D}\times\mathcal{D}} \end{cases}$$

such that there is at most one element $y \in \mathcal{D}$ for all $x \in \mathcal{D}$ with $(x, y) \in [\![p]\!]^{\mathcal{I}}$ for

Abstract form	Concrete form	Interpretation
$A_1 \| \ldots \| A_n$	(disjoint $A_1 \ldots A_n$)	$[\![A_i]\!]^{\mathcal{I}} \cap [\![A_j]\!]^{\mathcal{I}} = \emptyset, i \neq j$
$\{A_1, \ldots, A_n\} \bigcirc B$	(cover B $A_1 \ldots A_n$)	$\bigcup_{k=1}^{n} [\![A_k]\!]^{\mathcal{I}} \supseteq [\![B]\!]^{\mathcal{I}}$
$\{A_1, \ldots, A_n\} \oplus B$	(disjcover B $A_1 \ldots A_n$)	$\bigcup_{k=1}^{n} [\![A_k]\!]^{\mathcal{I}} \supseteq [\![B]\!]^{\mathcal{I}}$, $[\![A_i]\!]^{\mathcal{I}} \cap [\![A_j]\!]^{\mathcal{I}} = \emptyset, i \neq j$
$\|A\| \leq 1$	(indiv A)	$\|[\![A]\!]^{\mathcal{I}}\| \leq 1$
$A \rightarrow B$	(implies A B)	$[\![A]\!]^{\mathcal{I}} \subseteq [\![B]\!]^{\mathcal{I}}$

Table 4: CTL: additional restrictive operators

all $p \in \mathbf{p}$, and such that $[\![i]\!]^{\mathcal{I}} \neq [\![j]\!]^{\mathcal{I}}$, for all $i \neq j$. We will also use a functional notation for roles, i.e. $[\![R]\!]^{\mathcal{I}}(d) = \{e | (d,e) \in [\![R]\!]^{\mathcal{I}}\}$.

For example, given the concept symbol Workstation, intended to denote all workstations, and the role symbol disk-drive, intended to denote the disk drives on a given workstation, according to Table 1 the expression (Workstation $\sqcap \leq 0$ disk-drive) denotes all disk-less workstations. If we want to restrict the interpretation of the symbol Diskless-Workstation to the interpretation of (Workstation $\sqcap$ $\leq$ 0 disk-drive), the terminological axiom

$$\text{Diskless-Workstation} \doteq (\text{Workstation} \sqcap \leq 0 \text{ disk-drive})$$

can be used. According to Table 3, this axiom restricts the possible interpretations such that the symbol on the left hand side is interpreted as being equivalent to the expression on the right hand side.

Based on the set-theoretic semantics, the intuitive notion of *subsumption* between concepts that has been mentioned in the Introduction can be formalized as follows. A concept C is *subsumed* by a concept D relative to a set of terminological axioms and restrictions $\mathcal{T}$, written $C \preceq_{\mathcal{T}} D$, if the interpretation of C is necessarily included in the interpretation of D, i.e., $[\![C]\!]^{\mathcal{I}} \subseteq [\![D]\!]^{\mathcal{I}}$ for all possible interpretations $\mathcal{I} = \langle \mathcal{D}, [\![\cdot]\!]^{\mathcal{I}} \rangle$ that satisfy the restrictions specified by $\mathcal{T}$. If C is subsumed by D and D is also subsumed by C, we say C and D are *equivalent* and write $C \approx_{\mathcal{T}} D$. Finally, a concept C is said to be *incoherent* if it is semantically empty, i.e., if $C \approx_{\mathcal{T}} \bot$.

5 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.

5.1 Expressiveness of the Systems

First of all, the terminological languages that are supported by the various systems are quite different as can be seen from Tables 5–8.

	System					
	BACK	CLASSIC	KRIS	LOOM	MESON	SB-ONE
A	Y	Y	Y	Y	Y	Y
$C \sqcap D$	Y	Y	Y	Y	Y	Y
$C \sqcup D$			Y	Y		
$\neg C$			Y			
$\forall R{:}C$	Y	Y	Y	Y	Y	Y
$\exists R$	I	I	Y	Y	I	I
$\geq nR$	Y	Y	Y	Y	Y	Y
$\leq nR$	Y	Y	Y	Y	Y	Y
nR	I	I	I	Y	Y	I
$(= 0 \vee \geq n)R$			I	I		Y
$\exists R{:}C$			Y	Y		
$\geq nR{:}C$				I		
$\leq nR{:}C$				I		
$nR{:}C$				I		
$RC = SC$				Y	Y	Y
$RC \neq SC$				Y		Y
$RC \subseteq SC$				Y	Y	Y
$r : C$		Y	Y	Y		
$r : i$		Y		Y		
$rc = sc$				Y		
$rc \neq sc$				Y		
$rc \overset{\downarrow}{=} sc$		Y	Y	Y	I	I
$rc \overset{\downarrow}{\neq} sc$			Y	Y		I
$\{i_1, i_2, \ldots, i_m\}$	Y	Y		Y		

Y – explicitly present
I – implicitly present

Table 5: Expressiveness: concept forming operators

While three of the six systems use a similar syntactic scheme—similar to the one first used by Brachman and Levesque [9]—and one system adapted this syntactic scheme for PROLOG, i.e., using 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) operators varies, as well. Table 5 (Table 6) determines for every concept forming

	System					
	BACK	CLASSIC	KRIS	LOOM	MESON	SB-ONE
P	Y	Y	Y	Y	Y	Y
$R \sqcap S$	Y		Y	Y		
$R \sqcup S$						
R^{-1}				Y	Y	
r^{-1}				Y		
$R\|_C$	I			Y	Y	Y
$_C\|R$	I			Y		Y
$C \times D$	Y			I		Y
R^+						
RC				Y	Y	Y
r		Y	Y	Y		
$(R_1, \ldots, R_n)$				Y	Y	Y
1				Y		
p		Y	Y	Y		
$r\|_C$				Y		
rc		I	Y	Y		
$(r_1, \ldots, r_n)$		Y	Y	Y		
1				Y		

Y	– explicitly present
I	– implicitly present

Table 6: Expressiveness: role and attribute forming operators

operator (role and attribute forming operator) its explicit (marked by an 'Y') or implicit (marked by an 'I') existence in a system. Table 7 lists the types of terminological axioms that are supported by the systems. Finally, Table 8 lists the available additional restrictive operators.

In summary, the common intersection of all languages we considered is quite small. It contains only the concept-forming operators $C \sqcap D$ (*concept conjunction*), $\forall R\colon C$ (*value restriction*), and $\geq n\, R$ as well as $\leq n\, R$ (*number restrictions*) and the possibility to introduce *defined concepts* and *primitive concepts* using $A \doteq C$ and $A \sqsubseteq C$, respectively.

The differences in the syntactic surface form and expressiveness 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

	System					
	BACK	CLASSIC	KRIS	LOOM	MESON	SB-ONE
$A \doteq C$	Y	Y	Y	Y	Y	Y
$A \sqsubseteq C$	Y	Y	Y	Y	Y	Y
$P \doteq R$			Y	Y	Y†	Y
$P \sqsubseteq R$	Y		Y	Y	Y†	Y
$P \sqsubseteq \top$	Y	Y	Y	Y		Y
$p \doteq r$			Y	Y		
$p \sqsubseteq \top$		Y	Y	Y		

† – only inside concept definitions

Table 7: Expressiveness: terminological axioms

	System					
	BACK	CLASSIC	KRIS	LOOM	MESON	SB-ONE
$A_1 \Vert \ldots \Vert A_n$	Y	Y	I	I	Y	Y
$\{A_1, \ldots, A_n\} \bigcirc B$				I		Y
$\{A_1, \ldots, A_n\} \bigcirc\!\!\!\!\!\Vert\; B$				Y		I
$\lvert A \rvert \leq 1$	Y				Y	Y
$A \rightarrow B$		Y		Y	Y	

Y	– explicitly present
I	– implicitly present

Table 8: expressiveness: additional restrictive operators

of auxiliary concepts influences the runtime performance significantly—a point we will return to in Section 6.

5.2 Problematic Cases

Discounting the differences in syntax and expressiveness, one might expect that the common semantic framework—as spelled out by Brachman and Levesque [9] and sketched in the previous section—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 [26], 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 an error by the user. So, some systems mark the definitions of *semantically equivalent* concepts as an error or refuse to accept semantically empty (*incoherent*) concepts, for instance.

In the following, we use the abstract syntax to specify the tests we designed. Note that at some points the systems do not support the original language constructs, so we had to use a different formulation to check the system behavior. The alternative formulations are not always semantically equivalent. However, in the context where we used them they led to identical results. For instance, when concept disjointness $A \| B$ was not available, we used disjoint number restrictions to enforce the disjointness of the concepts as shown in Table 9.

Used Construct	Alternative Construct
$C \| D$	$C \doteq \geq 2R$, $D \doteq \leq 1R$
$\{C, D\} \bigcirc A$	$A \doteq C \sqcup D$
$C \doteq \geq nR{:}\, D$	$R_1 \sqsubseteq R$, $C \doteq (\geq nR_1) \sqcap (\forall R_1{:}\, D)$
$C \doteq \leq nR{:}\, D$	$R_1 \sqsubseteq R$, $C \doteq (\leq nR_1) \sqcap (\forall R_1{:}\, D)$
$C \doteq \forall R{:}\, (D \sqcap E)$	$C_1 \doteq \forall R{:}\, D$, $C_2 \doteq \forall R{:}\, E$, $C \doteq C_1 \sqcap C_2$

Table 9: Problematic cases: alternative constructs

We performed the following tests in order to get an idea how the systems deal with problematic cases:

1. How does the system handle *syntactically incorrect input*? Using a LISP-like notation, we checked what happened when keywords are misspelled or the expression is not well-formed:

   (a) *Input:* `(dfconcept a)`

   (b) *Input:* `(defconcept a b c)`

   (c) *Input:* `(defrole a (domain))`

   (d) *Input:* `(defconcept a (domain b))`

2. How does the system react to *incoherent* concepts, i.e., to concepts with a necessarily empty interpretation? Although this is completely legal in the semantic framework sketched above, and does not lead to the inconsistency

of the knowledge base as a whole, the system designer may choose to raise an error or output a warning message. Two cases have to be distinguished here. First, a *concept subexpression* may be incoherent, which does not lead necessarily to incoherency of the embedding expression. Second, a concept definition may result in an incoherent concept symbol, which is rather useless—and most probably an error. We used the following expressions to check out the system behavior:

(a) *Input:* $E \doteq (\geq 2R) \sqcap (\leq 1R)$
(b) *Input:* $C \| D,\ E \doteq (\forall R\colon (C \sqcap D))$

3. How are *semantically equivalent concept* definitions handled? Again, this is perfectly legal in the semantic framework, but the system designer may have decided to issue a warning because such equivalent definitions seem to be useless.

   (a) *Input:* $C \doteq\ \leq 1R,\ D \doteq C \sqcap (\geq 1R),\ E \doteq 1R$
   (b) *Input:* $C \doteq \forall R\colon D,\ E \doteq \forall R\colon D$

4. If a concept or role *symbol appears more than once* on the left hand side of a definition symbol, there are two options to handle this case. First, one may decide to prohibit this and issue an error message. Secondly, it is possible to interpret the second definition as a revision, i.e., the first definition is discarded and the second definition is used as the actual definition.

   (a) *Input:* $C \doteq\ \geq 1R,\ D \doteq\ \leq 1R,\ E \doteq C \sqcap D,\ C \doteq\ \geq 2R$
   Query: $E \approx_{\mathcal{T}} \bot$
   (b) *Input:* $B \| C,\ R \sqsubseteq \top \times D,\ E \doteq (\geq 1R) \sqcap (\forall R\colon B),\ R \sqsubseteq \top \times C$
   Query: $E \approx_{\mathcal{T}} \bot$

5. What kind of *general constraints* are enforced? Some systems require that no concept symbol is defined to be equivalent to the most general concept. Others enforce the restriction that if two concepts are declared to be disjoint, then they have to be *primitive concepts*, i.e., to specify only necessary but no sufficient conditions.

   (a) *Input:* $C \doteq \top$
   (b) *Input:* $F \doteq H,\ G \doteq I,\ F \| G$

6. Does the system accept *forward references* and the use of *previously undefined* concepts and roles? The semantic framework does not say anything about the linear order of concept and role definitions. Thus, the system designer has to choose whether all role and concept symbols have to be defined before they are used or whether forward definitions and "definitions by use" are permitted.

(a) *Input:* $D \doteq\ \geq 2R$, $C \doteq D \sqcap E$, $E \doteq\ \leq 1R$
Query: C $\preceq_{\mathcal{T}} \bot$

(b) *Input:* $C \| D$, $A \doteq \exists R{:}\, C$, $R \sqsubseteq B \times D$
Query: A $\preceq_{\mathcal{T}} \bot$

(c) *Input:* $C \doteq \exists R$, $S \sqsubseteq R$

7. If forward references are permitted, then it is possible to make circularly referring definitions. How does the system deal with them? The semantics given in this paper allows cycles, but there are other possible semantics for cyclic definitions [4, 15, 26]. Further, it is not a trivial task to provide algorithms that correspond to any of the possible semantics.

(a) *Input:* $C \doteq \forall R{:}\, C$

(b) *Input:* $C \doteq \forall R{:}\, C$, $D \doteq C \sqcap (\forall R{:}\, D)$
Query: $C \approx_{\mathcal{T}} D$

(c) *Input:* $C \doteq \forall R{:}\, D$, $D \doteq \forall R{:}\, C$
Query: $C \approx_{\mathcal{T}} D$

(d) *Input:* $C \doteq \exists R{:}\, D$, $D \doteq \exists R{:}\, C$
Query: $C \approx_{\mathcal{T}} D$
(result depends on underlying semantics)

8. How does the system fix the *syntactic category* of a symbol? It is possible to use the same symbol as a role and as a concept, disambiguating by context—if the concrete syntax supports that. Does the system permit such overloading of symbols?

(a) *Input:* $C \doteq \exists R{:}\, D$, $E \doteq \exists C$

The results of the tests of problematic cases described above are given in Table 10. The 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.

5.3 TBox Inferences

In order to get a feeling "how complete" the systems are, we designed a number of small test cases with more or less obvious conclusions. The tests are by no means exhaustive or systematic, but some of them were designed with a slightly

Result of Test ...		System					
		BACK	CLASSIC	KRIS	LOOM	MESON	SB-ONE
1	(a)	PErr	CErr	CErr	CErr	CErr	PErr
	(b)	PErr	CErr	CErr	CErr	CErr	CErr
	(c)	PErr	CErr	CErr	CErr	CErr	CErr
	(d)	—	CErr	CErr	CErr	CErr	CErr
2	(a)	Corr	Corr	Corr	Corr	CErr	CErr
	(b)	Corr	Corr	Corr	Corr	CErr	CErr
3	(a)	Corr	Corr	Corr	Corr	CErr	Corr
	(b)	Corr	Corr	Corr	Corr	CErr	Corr
4	(a)	Corr	CErr	Incorr	Corr	CErr	—
	(b)	Incorr	CErr	Incorr	—	—	—
5	(a)	Corr	CWarn	CWarn	CWarn	CErr	CWarn
	(b)	CErr	—	—	—	CErr	Corr
6	(a)	Corr	CErr	Corr	Corr	CErr	CErr
	(b)	Incorr	CErr	Corr	Corr	—	CErr
	(c)	Corr	CErr	Corr	Corr	Corr	CErr
7	(a)	CErr	CErr	CErr	(Corr)	(Corr)	(Corr)
	(b)	CErr	CErr	CErr	(Corr)	(Corr)	(Corr)
	(c)	CErr	CErr	CErr	(Corr)	CErr	(Corr)
	(d)	CErr	CErr	CErr	(Corr)	CErr	(Corr)
8	(a)	CErr	CErr	Incorr	Corr	—	Corr

PErr	error message of underlying programming system
CErr	case is error in system interpretation and an error message is issued
CWarn	case is no error in system interpretation but a warning is issued
Corr	case is no error in system interpretation and handled correctly without warning
(Corr)	system accepts input but does not classify these concepts according to one of the possible semantics
Incorr	case is not handled correctly
—	not applicable

Table 10: Problematic cases: test results

malicious attitude. As in Section 5.2, we sometimes used reformulation of the tests given below by using the correspondences of Table 9.

It turned out that some systems failed to draw the right conclusion even in cases where they are supposed to be complete. Since these results have been given to the system designers, these "bugs" are probably not present in more recent system versions. Nevertheless, these results demonstrate that it would be profitable to have an exhaustive and systematic test suite available on which all systems could be tested.

1. One basic inference is the detection of incoherency caused by disjoint concepts. We tested this for direct conjunctions (a), conjunctions appearing as value restrictions (b), and conjunctions of sub-concepts of two disjoint concepts (c). The case (d) may look a bit pathological, however, it is well-defined and easy to handle. Nevertheless, most systems failed on this example.

   (a) *Input:* $C \| D$, $E \doteq C \sqcap D$
   Query: E $\preceq_{\mathcal{T}} \bot$

   (b) *Input:* $C \| D$, $E \doteq \forall R{:}\,(C \sqcap D) \sqcap \exists R$
   Query: E $\preceq_{\mathcal{T}} \bot$

   (c) *Input:* $C \| D$, $E \sqsubseteq C$, $F \sqsubseteq D$
   Query: $E \| F$

   (d) *Input:* $C \sqsubseteq D$, $C \| D$
   Query: C $\preceq_{\mathcal{T}} \bot$

2. Another basic inference is to detect incoherence of a concept expression when the minimum and maximum restrictions ($\geq nR$ and $\leq nR$) on the same role are incompatible (a). A more complicated instance is case (b) where the disjointness of the concepts used in the existential quantification ($\exists R{:}\, C$) leads to the conclusion that there must be at least two role fillers. In fact, most systems are incomplete in this aspect that involves reasoning about the number of role fillers in the general case (see also test 3).

   (a) *Input:* $E \doteq (\geq 2R) \sqcap (\leq 1R)$
   Query: E $\preceq_{\mathcal{T}} \bot$

   (b) *Input:* $C \| D$, $E \doteq (\leq 1R) \sqcap (\exists R{:}\, C) \sqcap (\exists R{:}\, D)$
   Query: E $\preceq_{\mathcal{T}} \bot$

3. If a language contains minimum and maximum restrictions and sub-roles (i.e., the possibility of expressing role conjunctions) or qualified number restrictions ($\geq nR{:}\, C$ and $\leq nR{:}\, C$), then the reasoning about the number of potential role fillers can be become quite complicated. Case (a) is a comparably easy case resembling test 2 (b). Cases (b)–(e) are more complicated, but all inferences are based on the fact that because of disjointness and/or covering of role-filler concepts by another concept additional minimum or maximum restrictions for sub-roles (or qualified existential restrictions) could be derived.

   (a) *Input:* $C \| D$, $E \doteq (\geq 1R) \sqcap (\exists R{:}\, C) \sqcap (\exists R{:}\, D)$
   Query: $E \preceq_{\mathcal{T}} (\geq 2R)$

(b) *Input:* $C \| D,\ E \doteq (\leq 2R) \sqcap (\exists R\colon C) \sqcap (\exists R\colon D)$
Query: $E \preceq_{\mathcal{T}} (\leq 1R\colon C) \sqcap (\leq 1R\colon D)$

(c) *Input:* $\{C, D\} \bigcirc\!\!\!\!\!\!\parallel\; A,\ R \sqsubseteq T|_C,\ S \sqsubseteq T|_D,$
$E \doteq (\forall T\colon A) \sqcap (\geq 3T) \sqcap (\leq 1R) \sqcap (\leq 1S)$
Query: $\mathrm{E} \preceq_{\mathcal{T}} \bot$

(d) *Input:* $\{C, D\} \bigcirc A,\ E \doteq (\forall R\colon A) \sqcap (\geq 3R) \sqcap (\leq 1R\colon C)$
Query: $E \preceq_{\mathcal{T}} (\geq 2R\colon D)$

(e) *Input:* $R_1, R_2, R_3 \sqsubseteq R,\ T_1, T_2, T_3 \sqsubseteq T,\ C\|D\|E,$
$F_1 \doteq \exists R_1\colon ((\leq 2T) \sqcap (\exists T_1\colon C)))$
$F_2 \doteq \exists R_2\colon ((\leq 2T) \sqcap (\exists T_2\colon D)))$
$F_3 \doteq \exists R_3\colon ((\leq 2T) \sqcap (\exists T_3\colon E)))$
$F \doteq F_1 \sqcap F_2 \sqcap F_3$
Query: $F \preceq_{\mathcal{T}} (\geq 2R)$

4. The role-forming operator that restricts the range of a role ($R|_C$) leads to intractability in the general case [21]. In fact, this operator can be used to model "disjunctive reasoning." Case (a) is an example where *modus ponens* like reasoning is necessary. Case (b) shows an example where reasoning by case is required.

   (a) *Input:* $C \doteq (\forall R\colon D) \sqcap (\forall R|_D\colon E)$
   Query: $\mathrm{C} \preceq_{\mathcal{T}} (\forall R\colon E)$

   (b) *Input:* $C\|D,\ E \doteq (\forall R|_{(\geq\, 2S)}\colon C) \sqcap \forall R\colon D$
   Query: $E \preceq_{\mathcal{T}} \forall R\colon (\leq 1S)$

5. Role-value maps ($RC = SC$) are known to cause undecidability of subsumption [34]. However, even if role-value maps are handled in an incomplete manner, there are still some inferences that can be easily computed, for instance, incoherence caused by conflicting value, minimum or maximum restrictions, as in the cases (a)–(b), or caused by conflicting role-value map specifications, as in case (c).

   (a) *Input:* $E \doteq (1R) \sqcap (2S) \sqcap (R = S)$
   Query: $\mathrm{E} \preceq_{\mathcal{T}} \bot$

   (b) *Input:* $C\|D,\ E \doteq (\forall R\colon C) \sqcap (\forall S\colon D) \sqcap (R = S) \sqcap (\exists R)$
   Query: $\mathrm{E} \preceq_{\mathcal{T}} \bot$

   (c) *Input:* $C \doteq (\leq 1R) \sqcap (R = S),\ D \doteq (\leq 1R) \sqcap (R \neq S)$
   Query: $C\|D$

6. While equality reasoning over chains of roles is undecidable, equality reasoning over chains of attributes (i.e., single-valued roles) is quite easy [3] and

can be easily added to terminological languages [19]. Interestingly, the two systems that claimed to be complete in this aspect, namely CLASSIC and KRIS, failed on at least one of the tests given below.

(a) *Input:* $E \doteq (rst \overset{\downarrow}{=} uv) \sqcap (rstx \overset{\downarrow}{=} vu)$
Query: $E \preceq_{\mathcal{T}} rstx \overset{\downarrow}{=} uvx,\ E \preceq_{\mathcal{T}} uvx \overset{\downarrow}{=} vu$

(b) *Input:* $F \doteq (rst \overset{\downarrow}{=} uvw) \sqcap (rs \overset{\downarrow}{=} u)$
Query: $F \preceq_{\mathcal{T}} rsvw \overset{\downarrow}{=} ut$

7. Finally, we tested the role-forming operator *inverse* (R^{-1}).

(a) *Input:* $E \doteq \forall R\colon (\forall R^{-1}\colon A) \sqcap \exists R$
Query: $E \preceq_{\mathcal{T}} A$

The results of the tests are displayed in Table 11. Most of them simply confirm the formal or informal specification of the system. Below we summarize some interesting points:

BACK: The negative results for the tests 2 (b) and 3 (a)–(e) are not surprising since BACK does not compute the cardinality of role-filler sets of sub- and super-roles [24, 33]. The negative result for test 1 (d) is surprising, however.

CLASSIC: The interesting point is that CLASSIC fails on case 6 (a), although CLASSIC is supposed to be complete in this aspect.

KRIS: Even more surprising to us was the fact that the tested version of KRIS was incomplete for many examples. However, we tested one of the first prototypical versions. Further, we used one particular subsumption algorithm, namely, the $\mathcal{ALCFN}$-subsumption algorithm. When using the more powerful but slower $\mathcal{ALCFNR}$-subsumption algorithm, these errors did not occur. These cases are marked by 'N/Y' in the table.

LOOM: The incompleteness of LOOM was no big surprise to us since this system is explicitly described as supporting a very expressive terminological language in an incomplete manner. Some of the tests may actually lead to a positive result if "ABox"-reasoning is used. Since we tested only the terminological part of the system, this was not taken into account.

MESON: The incompleteness in case 4 (a) is no surprise. For the other two cases, a similar remark as above applies. The conclusions are drawn if "ABox"-reasoning is employed.

SB-ONE: The interesting point about SB-ONE is that it tries to account for reasoning with the cardinality of role-filler sets for sub- and super-roles, in an incomplete manner, though.

Result of Test ...		System					
		BACK	CLASSIC	KRIS	LOOM	MESON	SB-ONE
1	(a)	Y	Y	Y	Y	Y	Y
	(b)	Y	Y	N	Y	Y	Y
	(c)	Y	Y	Y	Y	Y	Y
	(d)	N	—	—	—	N	N
2	(a)	Y	Y	Y	Y	Y	Y
	(b)	N	—	Y	N	Y	Y
3	(a)	N	—	Y	N	Y	Y
	(b)	N	—	—	N	Y	N
	(c)	N	—	—	N	—	Y
	(d)	N	—	—	N	—	N
	(e)	N	—	N/Y	N	—	—
4	(a)	—	—	—	N	N	—
	(b)	—	—	—	N	—	—
5	(a)	—	—	—	Y	N	Y
	(b)	—	—	N/Y	Y	—	Y
	(c)	—	—	N/Y	N	—	N
6	(a)	—	N	N/Y	N	—	—
	(b)	—	Y	N/Y	N	—	—
7	(a)	—	—	—	N	Y	—

Y	inference drawn
N	inference not drawn
N/Y	result depends on the selected subsumption algorithm in KRIS

Table 11: TBox inferences: test results

5.4 Summary of Qualitative Results

The above results show that the ongoing process of specifying a common language for terminological representation and reasoning systems [28, 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.[1]

[1]Our set of tests and the tests developed for the BACK system [38] may be a starting point

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.

6 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 versions of BACK, CLASSIC, KRIS, and LOOM are available that are faster and/or support more expressive languages.

6.1 Hard Cases

Computational complexity analyses show that *subsumption determination* between *terms* is NP-hard [17] or even undecidable [34] 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 [25]. In order to explore this issue, we designed four tests to determine the behavior of the systems under conditions that are known to be hard.

The first three tests exploit the NP-hardness result for subsumption in terminologies [25], where the third test was designed in a way such that clever subsumption algorithms are also bound to use exponential time [25, p. 245].[2] The fourth test exploits the NP-hardness result for term-subsumption for languages that contain concept-conjunction, value restrictions, and qualified existential restrictions [16].

1. When computing the subsumption relation between C_n and D_n in the example below, some systems may expand the definitions leading to an expanded form that has a size exponential in the size of the original set of definitions.

for developing such a suite.

[2]Note that the example [25, p. 245] is not correct. The second appearance of each role R in all definitions should be R'.

Input:

$$\begin{aligned} &C_0 \sqsubseteq \top \\ &C_1 \doteq (\forall R_1 \colon C_0) \sqcap (\forall R_2 \colon C_0) \\ &C_2 \doteq (\forall R_1 \colon C_1) \sqcap (\forall R_2 \colon C_1) \\ &\qquad\vdots \\ &C_n \doteq (\forall R_1 \colon C_{n-1}) \sqcap (\forall R_2 \colon C_{n-1}) \end{aligned}$$

Concepts D_0, ..., D_n are defined analogously.

Query: $C_n \preceq_{\mathcal{T}} D_n$ for $n = 4, 8, 12$ (should fail)

2. The following test is similar to the above, but uses concept conjunctions instead of concept symbols as value restrictions.

Input:

$$\begin{aligned} &C_0 \sqsubseteq \top \\ &C_1 \sqsubseteq \top \\ &C_2 \doteq \forall R_1 \colon (C_0 \sqcap C_1) \\ &C_3 \doteq \forall R_2 \colon (C_0 \sqcap C_1) \\ &\qquad\vdots \\ &C_{2n} \doteq \forall R_1 \colon (C_{2n-2} \sqcap C_{2n-1}) \\ &C_{2n+1} \doteq \forall R_2 \colon (C_{2n-2} \sqcap C_{2n-1}) \end{aligned}$$

Concepts D_0, ..., D_{2n+1} are defined analogously.

Query: $C_{2n+1} \preceq_{\mathcal{T}} D_{2n+1}$ for $n = 4, 8, 12$ (should fail)

3. The next test is an example for the fact that in almost all systems one can get exponential answer times. We model a sequence of concepts in a way such that even if the best known algorithm is used (resembling an algorithm to decide equivalence for non-deterministic finite state automata) exponential time is necessary [25].

Input:

$$\begin{aligned} &C_{2n} \sqsubseteq \top \\ &C_i \doteq \begin{cases} (\forall R_1 \colon C_{i+1}) \sqcap (\forall R_2 \colon (C_{i+1} \sqcap C_{2i})) & \text{for } i = 0, \ldots, n \\ (\forall R_1 \colon C_{i+1}) \sqcap (\forall R_2 \colon C_{i+1}) & \text{otherwise} \end{cases} \end{aligned}$$

Concepts D_0, ..., D_{2n} are defined analogously.

Query: $C_0 \preceq_{\mathcal{T}} D_0$ for $n = 4, 8, 12$ (should fail)

4. The last test was inspired by the algorithms developed by Schmidt-Schauß and Smolka [35] and Donini *et al.* [17]. The restrictions in the "$\forall R$" parts have to be propagated to the two "$\exists R$" parts at the same nesting level (because they should hold for every R) and this leads to an exponential increase of the definition of the concepts C and D.

Input:

$$\begin{aligned}
E &\doteq \forall R\colon A\\
F &\doteq \forall R\colon B\\
G &\doteq \forall R\colon (A \sqcap B)\\
C &\doteq \ \exists R\colon E \sqcap \exists R\colon G \sqcap\\
&\quad \underbrace{\begin{array}{l}
\forall R\colon (G \sqcap \exists R\colon E \sqcap \exists R\colon G \sqcap\\
\qquad \forall R\colon (G \sqcap \exists R\colon E \sqcap \exists R\colon G \sqcap\\
\qquad\qquad \forall R\colon (G \sqcap \exists R\colon E \sqcap \ldots\\
\qquad\qquad\qquad \vdots\\
\qquad\qquad\qquad\quad \forall R\colon (G \sqcap \exists R\colon E \sqcap \exists R\colon G) \ldots)))
\end{array}}_{n \text{ levels of nesting}}
\end{aligned}$$

Concept D is defined analogously using F instead of E.

Query: $C \preceq_{\mathcal{T}} D$ for nesting depth $n = 4, 8, 12$.

The runtime performance of the different systems on the examples described above are displayed in Table 12. It should be noted, however, that 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.

The results of tests 1 and 2 clearly indicate that most systems do not expand concept definitions before checking subsumption. Only KRIS and SB-ONE (in test 2) seem to do that, leading to a rapidly growing runtime behavior in this case. Test 3 shows that all systems can be forced to exhibit an rapidly growing runtime behavior—even those systems that are announced to be incomplete and fast.

Finally, test 4 turned out to be a test that most systems were not able to deal with. It was not possible to express the test example in CLASSIC, MESON, and SB-ONE. Of the remaining three systems, BACK ran into an internal error, LOOM accepted the input, but did not detect the subsumption relation, and only KRIS computed the correct result—albeit using a rapidly growing runtime behavior.

6.2 Real Knowledge Bases

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

Result (sec) of Test ...		System					
		BACK*	CLASSIC°	KRIS°	LOOM°	MESON*	SB-ONE°
1	n=4	1	2	3	1	1	11
	n=8	1	4	77	3	2	33
	n=12	2	5	2680	5	6	56
2	n=4	7	3	82	7	3	99
	n=8	32	11	1867	22	23	859
	n=12	75	16	—†	39	84	3263
3	n=4	25	4	459	28	29	372
	n=6	352	40	18230	155	5099	1836
	n=8	6035	706	—†	666	—†	9500
4	n=4	—‡	—	4	4	—	—
	n=8	—	—	49	8	—	—
	n=12	—	—	745	13	—	—

*measured on Solbourne 601/32.

°measured on MacIvory.

†test has been aborted

‡system reported internal error

Table 12: Hard cases: runtime of subsumption tests

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. Below we give a brief description of these "real" knowledge bases:

CKB (Conceptual Knowledge Base): Contains knowledge about tax regulations and is used in the Natural Language project XTRA at the University of Saarbrücken.

Companies: Contains knowledge about company structures and is used at the Technical University Berlin in the framework of the ESPRIT project ADKMS.

FSS (Functional Semantic Structures): Contains knowledge about speech acts and is used in the Natural Language project XTRA at the University of Saarbrücken.

Espresso: Contains knowledge about Espresso machines and their structure. It is used in the WIP-Project of DFKI in the framework of multimodal presentation of information.

Wisber: Contains knowledge about different forms of investments and was used in the natural language dialog project WISBER at the University of Hamburg.

Wines simple kosher: Contains knowledge about wines, wineries, and meal-courses. It is used as at AT&T Bell Labs. as a sample KB for the CLASSIC system.[3]

Table 13 characterizes the structure of the original KBs by means of the number of defined and primitive concepts and roles, respectively. As mentioned above, by first manually translating the knowledge bases into the "common terminological language" and then translating them to each target language using our (semi-) automatic translators, some artificial concepts have been introduced, the cardinality is also shown in Table 13. The exact number of auxiliary concepts may differ from system to system, though.

Name	Original Language	defined	primitive	artificial	$\sum$	defined	primitive
		concepts				roles	
CKB	SB-ONE	23	57	58	138	2	46
Companies	BACK	70	45	81	196	1	39
FSS	SB-ONE	34	98	75	207	0	47
Espresso	SB-ONE	0	145	79	224	11	41
Wisber	TURQ	50	81	152	283	6	18
Wines	CLASSIC	50	148	237	435	0	10

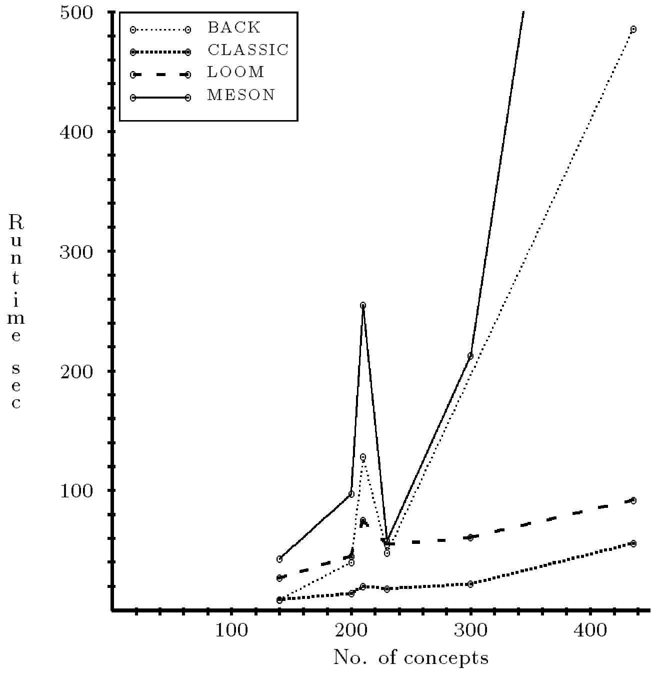
Table 13: Real knowledge bases: structural description

In Figure 2, the runtime for classifying the knowledge bases is plotted against the number of concepts defined in the different knowledge bases. The number of concepts were counted *after* the translation, i.e., this number includes the auxiliary concepts introduced in the translation process.

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 used two diagrams.

Second, the diagrams 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, $45,000/56 \approx 800$. It should be noted that this result

[3]A lot of individuals have been transformed to general concepts because in our tests we only considered terminological knowledge.



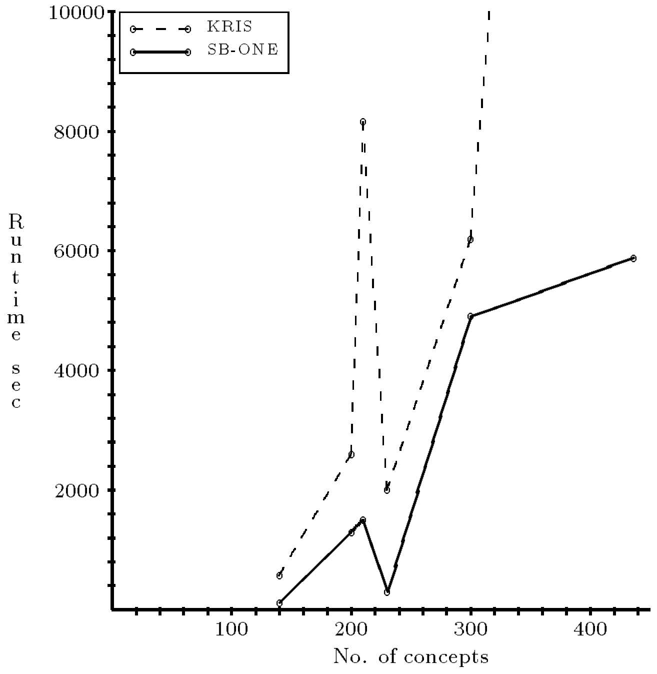


Figure 2: Runtime performance of classification for realistic KBs

cannot be attributed to the incompleteness of CLASSIC, the approximate character of the translation, or the differing number of auxiliary concepts introduced in the translation process. The knowledge base was formulated in a terminological language such that both CLASSIC and KRIS are complete, which implies that in this case the translation was also meaning preserving. Further, the knowledge base for KRIS contains fewer auxiliary concepts than the knowledge base for CLASSIC.

Considering that KRIS was developed as an experimental test bed 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 FSS knowledge base seems to be somehow special because the runtime curves show 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 our results concerning the runtime behavior under worst-case conditions (see Section 6.1), it is an indication that under realistic conditions the runtime behavior can be unexpectedly influenced by the structure of the knowledge base.

6.3 Random Knowledge Bases

Summarizing the curves in Figure 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 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 instance, running CLASSIC on the original Wine knowledge base (without the ABox) and the translated KB revealed that the second KB needed twice as much time as the first one. This might sound surprising since the translation is meaning preserving in this case. However, taking into account that the translation process introduces a large number of auxiliary concepts, this observation demonstrates that the number of concepts defined in a knowledge base is an important parameter for the runtime.

In order to eliminate the inaccuracy of measurement introduced by the translation process, to get an idea how the runtime varies with the number of concepts, and to test the systems on larger knowledge bases, a number of terminological knowledge bases were randomly generated using only the intersection of all terminological languages used in the systems (i.e., only conjunction, value restrictions, and number restrictions)—avoiding the translation problem. The structure of these generated knowledge bases resembles some of the aspects of the real knowledge bases we used (percentage of defined concepts, average number of declared super-concepts, average number of role restrictions, etc.). We do not claim, however, that the generated knowledge bases are realistic in all aspects.

The generated knowledge bases have the following properties:

- 80% of the concepts are "primitive" (i.e., introduced by $\sqsubseteq$).
- There are exactly 10 different roles.
- Each concept definition is a conjunction containing
  - one or two concept symbols (explicit super-concepts),
  - zero or one minimum restrictions,
  - zero or one maximum restrictions,
  - and zero, one, or two value restrictions,

  where the number of constructs from one category and the roles and concepts are randomly assigned with a uniform distribution. Further, the concepts are constructed in a way such that no concept is incoherent (i.e., no minimum restriction is larger than any maximum restriction).

In order to avoid forward references and definitional cycles, the concepts are partitioned into *layers*, where the ith layer has 3^i concepts. When assigning explicit super-concepts or value-restriction concepts to the definition of a concept from level i, only concepts from level 0 to $i-1$ are considered. The results of this test are given in Figures 3, 4, and 5.

Comparing the curves in these three figures with the curves in Figure 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 (personal communication), 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 held constant (same number of concepts and role restrictions).

The curves in Figures 3–5 indicate that the runtime grows faster than linearly with the number of concepts. We conjecture that in general the runtime of classification in 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, since it is not known how

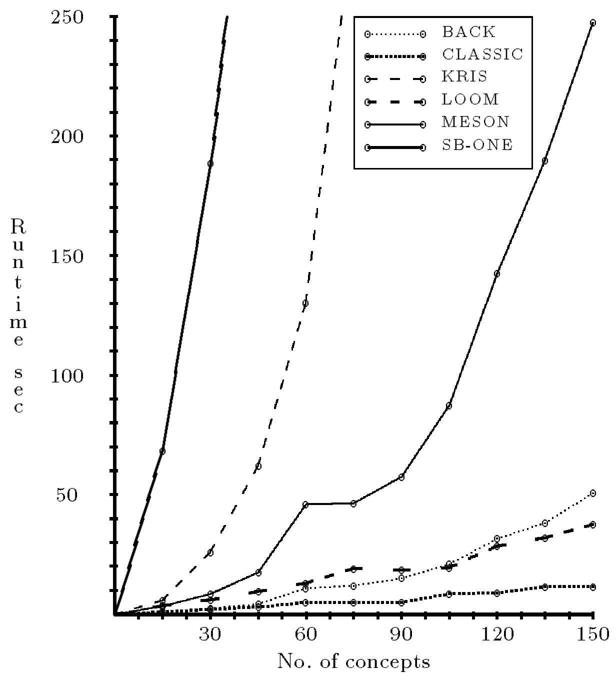


Figure 3: Runtime performance of classification for small random KBs

many partial orders exist for a given number of elements [2, p. 271], average case results are probably very hard to obtain except for special cases. For instance, if the subsumption hierarchy forms a tree, classification can be performed in $O(n \times \log n)$. Further, if the subsumption hierarchy deviates only slightly from a tree-structure, classification may still be done in $O(n \times \log n)$. However, the realistic knowledge bases we considered did not have such a structure.

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, together with the WINO group we were able to speed up the KRIS system significantly. Classification in KRIS is now almost as fast as in LOOM and CLASSIC [7] on the test set considered in this paper.

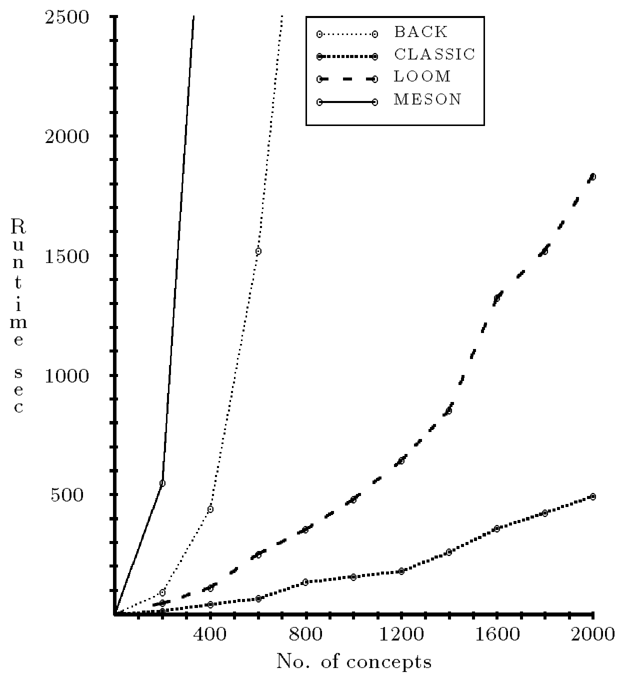


Figure 4: Runtime performance of classification for large random KBs

Finally, it should be noted that the results of our performance tests should not be taken for establishing a ranking between the systems. First of all, as mentioned already in Section 6.1, MESON and BACK are not directly comparable with the other systems because MESON and BACK were tested on a different, slightly faster machine. Second, most of the systems were probably not optimized for speed, and using profiling techniques it is relatively easy to speed up large LISP programs by a factor of two or three. Third, efficiency is only one dimension in evaluating a system. Nevertheless, we believe that our tests give an idea of how fast existing terminological representation systems are and how many concepts can be dealt with in reasonable time.

7 Conclusions

We have analyzed six different terminological representation and reasoning systems from a qualitative and quantitative point of view.[4] 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

[4]The CTL source code of the tests described in this paper is available via anonymous ftp from `duck.dfki.uni-sb.de` in the directory `pub/papers` as `RR-92-16.tests.tar.Z`.

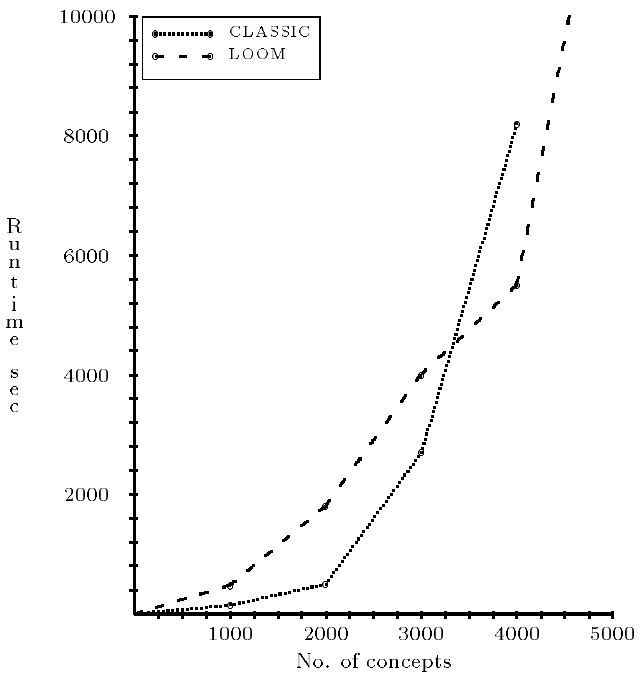


Figure 5: Runtime performance of classification for very large random KBs

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.

Acknowledgment

We would like to thank the members of the research groups KIT-BACK at the Technical University of 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

[1] *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, Boston, MA, Aug. 1990. MIT Press.

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

[3] H. Aït-Kaci. *A Lattice-Theoretic Approach to Computations Based on a Calculus of Partially Ordered Type Structures.* PhD thesis, University of Pennsylvania, Philadelphia, PA, 1984.

[4] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In AAAI-90 [1], pages 621–626.

[5] F. Baader, H.-J. Bürckert, J. Heinsohn, B. Hollunder, J. Müller, B. Nebel, W. Nutt, and H.-J. Profitlich. Terminological Knowledge Representation: A proposal for a terminological logic. DFKI Technical Memo TM-90-04, Saarbrücken, 1990. A revised version has been published in [27].

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

[7] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems or "making KRIS get a move on". In B. Nebel, W. Swartout, and C. Rich, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference*, pages 270–281, Cambridge, MA, Oct. 1992. Morgan Kaufmann.

[8] T. Bollinger and U. Pletat. The LILOG knowledge representation system. *SIGART Bulletin*, 2(3):22–27, June 1991.

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

[10] R. J. Brachman, V. Pigman Gilbert, and H. J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts in KRYPTON. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 532–539, Los Angeles, CA, Aug. 1985.

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

[12] R. Cattoni and E. Franconi. Walking through the semantics of frame-based description languages: A case study. In *Proceedings of the Fifth International Symposium on Methodologies for Intelligent systems*, Knoxville, TN, Oct. 1990. North-Holland.

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

[14] P. T. Devanbu, R. J. Brachman, P. G. Selfridge, and B. W. Ballard. LaSSIE: a knowledge-based software information system. *Communications of the ACM*, 34(5):35–49, May 1991.

[15] R. Dionne, E. Mays, and F. J. Oles. A non-well-founded approach to terminological cycles. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 767–773, San Jose, CA, July 1992. MIT Press.

[16] F. M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. M. Spacamella. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53(2-3):309–327, 1992.

[17] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J. A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, pages 151–162, Cambridge, MA, Apr. 1991. Morgan Kaufmann.

[18] B. R. Gaines. Empirical investigations of knowledge representation servers: Design issues and application experience with KRS. *SIGART Bulletin*, 2(3):45–56, June 1991.

[19] B. Hollunder and W. Nutt. Subsumption algorithms for concept languages. DFKI Research Report RR-90-04, German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, 1990.

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

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

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

[23] E. Mays, R. Dionne, and R. Weida. K-Rep system overview. *SIGART Bulletin*, 2(3):93–97, June 1991.

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

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

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

[27] B. Nebel, C. Peltason, and K. von Luck, editors. *International Workshop on Terminological Logics*, Schloß Dagstuhl, May 1991. Published as a KIT Report, TU Berlin, IWBS Report, IBM Germany, Stuttgart, and DFKI Document, DFKI, Kaiserlautern & Saarbrücken.

[28] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *The AI Magazine*, 12(3):36–56, 1991.

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

[30] P. F. Patel-Schneider, D. L. McGuinness, R. J. Brachman, L. Alperin Resnick, and A. Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113, June 1991.

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

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

[33] C. Peltason, A. Schmiedel, C. Kindermann, and J. Quantz. The BACK system revisited. KIT Report 75, Department of Computer Science, Technische Universität Berlin, Berlin, Germany, Sept. 1989.

[34] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In R. Brachman, H. J. Levesque, and R. Reiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 1st International Conference*, pages 421–431, Toronto, ON, May 1989. Morgan Kaufmann.

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

[36] J. G. Schmolze and W. S. Mark. The NIKL experience. *Computational Intelligence*, 6:48–69, 1991.

[37] N. K. Sondheimer and B. Nebel. A logical-form and knowledge-base design for natural language generation. In *Proceedings of the 5th National Conference of the American Association for Artificial Intelligence*, pages 612–618, Philadelphia, PA, Aug. 1986.

[38] J. Thomson. Semantic validation of inferences in terminological representation systems. KIT Report 90, Department of Computer Science, Technische Universität Berlin, Berlin, Germany, Apr. 1991.

[39] W. Wahlster, E. André, S. Bandyopadhyay, W. Graf, and T. Rist. WIP: the coordinated generation of multimodal presentations from a common representation. In A. Ortony, J. Slack, and O. Stock, editors, *Computational Theories of Communication and their Applications*, pages 121–144. Springer-Verlag, Berlin, Heidelberg, New York, 1992.

[40] W. Wahlster, E. André, W. Finkler, H.-J. Profitlich, and T. Rist. Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 62(3), 1993. To appear.

[41] B. C. Williams. Interaction-based invention: Designing novel devices from first principles. In AAAI-90 [1], pages 349–356.