Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-93-03

An Empirical Analysis of Optimization Techniques for Terminological Representation Systems

or: 'Making KRIS get a move on'

Franz Baader, Bernhard Hollunder, Bernhard Nebel,
Hans-Jürgen Profitlich, Enrico Franconi

January 1993

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

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

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

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- ❑ Intelligent Engineering Systems
- ❑ Intelligent User Interfaces
- ❑ Computer Linguistics
- ❑ Programming Systems
- ❑ Deduction and Multiagent Systems
- ❑ Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Friedrich J. Wendl
Director

An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or: 'Making KRIS get a move on'

Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi

DFKI-RR-93-03

A shorter version of this report has been published in the Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR-92), Cambridge, MA, 1992.

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW -8901 8).

An Empirical Analysis of Optimization Techniques for Terminological Representation Systems

or: Making KRIS get a move on*

Franz Baader, Bernhard Hollunder,
Bernhard Nebel, Hans-Jürgen Profitlich
German Research Center for AI (DFKI)
Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, Germany
e-mail: ⟨*last name*⟩@dfki.uni-sb.de

and

Enrico Franconi
Istituto per la Ricerca
Scientifica e Tecnologica (IRST)
38050 Povo TN, Italy
e-mail: franconi@irst.it

Abstract

We consider different methods of optimizing the classification process of terminological representation systems, and evaluate their effect on three different types of test data. Though these techniques can probably be found in many existing systems, until now there has been no coherent description of these techniques and their impact on the performance of a system. One goal of this paper is to make such a description available for future implementors of terminological systems. Building the optimizations that came off best into the KRIS system greatly enhanced its efficiency.

---

*This is a revised and extended version of a paper presented at the *3rd International Conference on Principles of Knowledge Representation and Reasoning*, October 1992, Cambridge, MA.

Contents

List of Figures

List of Tables

1 Introduction

Terminological representation systems can be used to represent the taxonomic and conceptual knowledge of a problem domain in a structured and well-formed way. To describe this kind of knowledge, one starts with atomic concepts (unary predicates) and roles (binary predicates), and defines more complex concepts and roles using the operations provided by the concept language of the particular formalism. In addition to this concept description formalism, most terminological representation systems also have an assertional component, which can be used to express facts about a concrete world.

Of course, it is not enough to have a system that just stores concept definitions and assertional facts. The system must also be able to reason about this knowledge. An important inference capability of a terminological representation system is *classification*. The classifier computes all *subsumption* relationships between concepts, i.e., the subconcept-superconcept relationships induced by the concept definitions. In this paper we consider only optimizations for the classification process. We do not take into account problems that are specific to assertional reasoning. This concentration on the terminological component is partially justified by the fact that this is the part that partakes in most reasoning activities of almost all systems—which means that the efficiency of this reasoning component is crucial for the overall behavior of the system. In addition, the only existing empirical analysis comparing the efficiency of different terminological systems is also restricted to the terminological part of the systems [Heinsohn *et al.*, 1992].

The first terminological representation system, KL-ONE [Brachman and Schmolze, 1985], was an implementation of Brachman's work on structured inheritance networks [Brachman, 1977]. In the last decade many knowledge representation systems based on these ideas have been built, for example BACK [Peltason, 1991], CLASSIC [Patel-Schneider *et al.*, 1991], KANDOR [Patel-Schneider, 1984], KL-TWO [Vilain, 1985], K-Rep [Mays *et al.*, 1991], KRYPTON [Brachman *et al.*, 1985], KRIS [Baader and Hollunder, 1991], LOOM [MacGregor, 1991], MESON [Edelmann and Owsnicki, 1986], NIKL [Schmolze and Mark, 1991], SB-ONE [Kobsa, 1991], and YAK [Cattoni and Franconi, 1990]. Moreover, formal aspects of terminological representation languages have been thoroughly investigated, with the highest emphasis having been placed on the decidability and complexity of the subsumption problem (see, e.g., [Levesque and Brachman, 1987; Nebel, 1988; Schmidt-Schauß, 1989; Patel-Schneider, 1989; Nebel, 1990b; Schmidt-Schauß and Smolka, 1991; Donini *et al.*, 1991a; Donini *et al.*, 1991b]). As a result of these investigations, it is known that subsumption determination is at least NP-hard or even undecidable for reasonably expressive languages. The developers of terminological representation systems usually have reacted to this problem in one of the following two ways. On the one hand, there are systems such as CLASSIC which support only a very limited terminological language, but employ almost complete reasoning methods. On the other hand, systems such as LOOM provide for a very powerful language, but the reasoning is incomplete, which means that not all existing subsumption relationships are detected.

The only system that does not make this compromise, i.e., that provides complete algorithms for a very expressive concept description language, is KRIS. Obviously, this means that KRIS will need exponential time for worst case examples which, on the one hand, are not expressible in the less expressive systems, and which are, on the other hand, treated more efficiently, but less completely, by systems with fast and incomplete algorithms. However, it is not *a priori* clear whether this also implies that KRIS has to be less efficient for "typical" knowledge bases. In particular, it might at least be fast in cases where its full expressive power is not used, or where incomplete algorithms are still complete. The empirical analysis of terminological representation systems described in [Heinsohn *et al.*, 1992] seems to preclude this possibility, though. KRIS turned out to be much slower than, for example, CLASSIC, even for knowledge bases that are in the scope of CLASSIC's concept language, and for which CLASSIC's subsumption algorithm is complete.

One aim of the present paper is to demonstrate that this bad performance of KRIS is not mainly due to the use of complete subsumption algorithms, but instead to the fact that the tested version was the first implementation of an experimental system where efficiency considerations only played a minor role. For this purpose we shall consider possible optimizations of the classification process on three different levels. The optimizations on the *highest level* are independent of the fact that what we are comparing are concepts defined by a terminological language. On this level, classification is considered as the abstract order-theoretic problem of computing a complete representation of a partial ordering (in our case the subsumption hierarchy) by making as few as possible explicit comparisons (in our case calls of the subsumption algorithm) between elements of the underlying set (in our case the set of all concepts occurring in the terminology). Optimizations on the *next level* still leave the subsumption algorithm unchanged, but they do employ the fact that we are not comparing abstract objects but instead structured concepts. At this level subsumption relationships that are obvious consequences of this structure can be derived without invoking the subsumption algorithm. On the *third level*, the actual subsumption algorithm is changed so that it can benefit from the information provided by subsumption relationships which have previously been computed. The effects these optimizations have on the classification process are evaluated on three different sets of test data, which are described in Section 2 below.

It should be noted that we do not claim that all the presented optimizations are novel. Similar optimizations can probably be found in many of the existing systems (see, e.g., [Lipkis, 1982; MacGregor, 1988; Peltason *et al.*, 1989; Woods, 1991]). Further, the optimizations on the first level described below are very similar to methods that can be found in the conceptual graphs literature (see, e.g., [Levinson, 1984; Ellis, 1991; Levinson, 1992]), and which have been used in the implementation of the PEIRCE system [Ellis and Levinson, 1992]. However, until now it was not possible to find an exhaustive and coherent description of all the methods, and there were no empirical studies on their exact effects. A second motivation for this work is to make such a description available for future implementors of terminological representation systems.

2 The Test Data

In order to evaluate the different optimization techniques empirically, we used three sets of test data. As in [Heinsohn *et al.*, 1992], we considered both existing knowledge bases used in other projects (six different KBs with the number of concepts ranging between 140 and 440), and randomly generated knowledge bases whose structure resembles those of the six real knowledge bases.

First we give a brief description of the six realistic knowledge bases. Table 1 characterizes the structure of the original KBs by means of the number of defined and primitive concepts and roles, respectively. As mentioned in [Heinsohn *et al.*, 1992], in the process of automatically translating and adapting the KBs, some artificial concepts are introduced. The number of these concepts is also shown in Table 1. A more structural characterization of the subsumption hierarchy induced by the KBs is given in Table 2 in Section 3.4.

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 sample KB of the CLASSIC system.[1]

Name	defined	primitive	artificial	$\sum$	defined	primitive
	concepts				roles	
CKB	23	57	104	184	2	46
Companies	70	45	126	241	1	39
FSS	34	98	122	254	0	47
Espresso	0	145	124	269	11	41
Wisber	50	81	199	330	6	18
Wines	50	148	282	480	0	10

Table 1: Real Knowledge Bases: Structural description

In order to get an idea how the runtime performance varies with the number of concepts, and to test the optimizations on larger knowledge bases, a number of terminological knowledge bases were randomly generated using a minimal terminological languages containing only *concept conjunction, value restrictions, and number restrictions.* The structure of these generated knowledge bases resembles some of the aspects of the real knowledge bases we used. 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., the definition of the concept gives only the necessary conditions.
- 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,

[1] A lot of individuals have been transformed to general concepts because in our tests we only considered terminological knowledge but did not want to cut all the nice information about different wineries and wines.

– 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 inconsistent (i.e., no minimum restriction is larger than any maximum restriction).

In order to avoid definitional cycles, the concepts are partitioned into *layers*, where the ith layer has 3^i concepts. When assigning explicit superconcepts or value-restriction concepts to the concept definition of a concept from level i, only concepts from level 0 to $i-1$ are considered.

Comparing the randomly generated knowledge bases with real knowledge bases, one notes that the number of roles might not be realistic. Further, the randomly generated knowledge bases tend to have a concept hierarchy that is less tree-like than real knowledge bases. Nevertheless, in the empirical analysis of different terminological representation system [Heinsohn *et al.*, 1992], the runtime performance on the generated knowledge bases is similar to the runtime performance on real knowledge bases.

Since the first level of optimizations can be done in an abstract order-theoretic setting, these optimizations are also evaluated on randomly generated partial orderings [Winkler, 1985]. The generation process goes as follows. In order to generate a partial order $(\{1, \ldots, n\}, <_P)$:

1. Choose a positive integer k.
2. Randomly generate k permutations $\pi_i = (p_{1,i}, \ldots p_{n,i})$ on $\{1, \ldots, n\}$. Such a permutation defines a linear ordering $<_i$ on $\{1, \ldots, n\}$ as follows: $r <_i s$ iff r comes before s in π_i.
3. The strict partial ordering relation $<_P$ on $\{1, \ldots, n\}$ is now defined as: $r <_P s$ iff $r <_i s$ for all $i, 1 \leq i \leq k$.

Note that for $k = 1$, the resulting partial order is a total order. Further, for k approaching n, the generated partial orders tend to become flat, i.e., most elements will be pairwise incomparable.

3 Computing the Subsumption Hierarchy

In the first level of optimizations we are concerned with computing the concept hierarchy induced by the subsumption relation. More abstractly, this task can be viewed as computing the representation of a partial ordering. For a given partial ordering[2] $\leq$ on some set P, $\prec$ shall denote the *precedence*

[2] A partial ordering is a transitive, reflexive, and antisymmetric relation.

relation of $\leq$, i.e., $\prec$ is the smallest relation such that its reflexive, transitive closure is identical with $\leq$. Obviously, $x \prec y$ iff $x \leq y$ and there is no z different from x and y such that $x \leq z \leq y$. If $x \leq y$, we say that x is a *successor* of y and y is a *predecessor* of x. Similarly, if $x \prec y$, we say that x is an *immediate successor* of y and y is an *immediate predecessor* of x.

Given a set X and a partial ordering $\leq$ on X, computing the representation of this ordering on X amounts to identifying $\prec$ on X. If $\leq$ is a *total* ordering, this task is usually called sorting. For a partial ordering it is called the identification problem (see, e.g., [Faigle and Turán, 1988]). The basic assumption here is that the partial ordering is given via a comparison procedure, and that the comparison operation is rather expensive. For this reason, the complexity of different methods to compute the precedence relation is measured by counting the number of comparisons. Of course, the number of other operations should not be too high as well.

In our case, X is the set of concepts defined in a terminological knowledge base, and $\leq$ is the subsumption relation between these concepts. The assumption that the subsumption test is the most expensive operation is justified by the known complexity results for the subsumption problem [Donini *et al.*, 1991a]. To be more precise, the subsumption relation is only a quasi-ordering, i.e., it need not be antisymmetric. For the following discussion, this is mostly irrelevant, however. There is only one place in the algorithms where this fact has to be taken into account.

The worst case complexity of computing the representation of a partial ordering on a set with n elements is obviously $O(n^2)$ because it takes $n \times (n-1)$ comparisons to verify that a set of n incomparable elements is indeed a flat partial order. Since subsumption hierarchies typically do not have such a "pathological" structure, considerably less than $n \times (n-1)$ comparisons will almost always suffice.

Below, we describe and analyze four different methods to identify the representation of a partial ordering, namely, the *brute force* method, the *simple traversal* method, the *enhanced traversal* method, and the *chain inserting* method. Average case analyses of these methods seem to be out of reach since one does not know enough about the structure of "typical" terminological knowledge bases, and since it is not even known how many different partial orders exist for a given number of elements [Aigner, 1988]. For this reason, the different methods are compared empirically.

All methods we describe are incremental, i.e., assuming that we have identified the precedence relation $\prec_i$ for $X_i \subseteq X$, the methods compute for some element $c \in X - X_i$ the precedence relation $\prec_{i+1}$ on $X_{i+1} = X_i \cup \{c\}$. The two most important parts of this task are the *top search* and the *bottom search*. The top search identifies the *set of immediate predecessors* in X_i for a given element c, i.e., the set $X_i{\downarrow}c := \{x \in X_i \mid c \prec x\}$. Symmetrically, the bottom search identifies the *set of immediate successors* of c, denoted by

$X_i{\uparrow}c$.

To be more precise, the procedures for top search that we will describe below compute the set $\{x \in X_i \mid c \leq x \text{ and } c \not\leq y \text{ for all } y \prec_i x\}$, which in most cases is the set $X_i{\downarrow}c$. Because the subsumption relation is only a quasi-ordering, there is one exception. The concept c can be equivalent to an element x of X_i, i.e., $c \leq x$ and $x \leq c$. In this case, the top search procedures will yield $\{x\}$ instead of $X_i{\downarrow}c$. To take care of this case, we test $x \leq c$ whenever the top search procedure yields a singleton set $\{x\}$. If this test is positive, c is equivalent to x, and we know that $X_i{\downarrow}c = X_i{\downarrow}x$, and $X_i{\uparrow}c = X_i{\uparrow}x$, which means that we don't need the bottom search phase. Otherwise, the result of the top search procedure is in fact $X_i{\downarrow}c$.

Given $X_i{\downarrow}c$, $X_i{\uparrow}c$, and $\prec_i$, it is possible to compute the precedence relation $\prec_{i+1}$ on $X_{i+1} = X_i \cup \{c\}$ in linear time. In fact, one just has to add $\prec$-links between c and each element of $X_i{\downarrow}c$, and between each element of $X_i{\uparrow}c$ and c. In addition, all $\prec$-links between elements of $X_i{\uparrow}c$ and $X_i{\downarrow}c$ have to be erased.

3.1 The Brute Force Method

The top search part of the brute force method can be described as follows:

1. Test $c \leq x$ for all $x \in X_i$.
2. $X_i{\downarrow}c$ is the set of all $x \in X_i$ such that the test succeeded and for all $y \prec_i x$ the test failed.

The bottom search is done in the dual way.

This method obviously uses $2 \times |X_i|$ comparisons for the step of inserting c in X_i. Summing over all steps leads to $n \times (n-1)$ comparison operations to compute the representation of a partial ordering for n elements. Further, this is not only the worst-case, but also the best-case complexity of this method.

3.2 The Simple Traversal Method

It is obvious that many of the comparison operations in the brute force method can be avoided. Instead of testing the new element c blindly with all elements in X_i, in the top search phase the partial ordering can be traversed top-down and in the bottom search phase bottom-up, stopping when immediate predecessors or successors have been found. This leads us to the specification of the simple traversal method (see Figure 1).

The top search starts at the top[3] of the already computed hierarchy. For each concept $x \in X_i$ under consideration it determines whether x has an

[3] We assume that our concept hierarchies always contain a top element $\top$ and a bottom element $\bot$.

```
top-search(c,x) =
    mark(x,"visited")
    Pos-Succ ← ∅
    for all y with y ≺_i x do
        if simple-top-subs?(y,c)
          then Pos-Succ ← Pos-Succ ∪ {y}
        fi
    od
    if Pos-Succ is empty
      then return {x}
      else Result ← ∅
           for all y ∈ Pos-Succ do
               if not marked?(y,"visited")
                 then Result ← Result ∪ top-search(c,y)
               fi
           od
           return Result
    fi

simple-top-subs?(y,c) =
    if marked?(y,"positive")
      then return true
      elsif marked?(y,"negative")
        then return false
        elsif subs?(y,c)
          then mark(y,"positive")
               return true
          else mark(y,"negative")
               return false
        fi
      fi
    fi
```

Figure 1: Top search phase of the simple traversal method

immediate successor y satisfying $c \leq y$. If there are such successors, they are considered as well. Otherwise, x is added to the result list of the top search.

In order to avoid multiple visits of elements of X_i and multiple comparisons of the same element with c, the top search algorithm described in Figure 1 employs one label to indicate whether a concept has been "visited" or not and another label to indicate whether the subsumption test was "positive," "negative," or has not yet been made. The procedure *top-search* gets two concepts as input: the concept c, which has to be inserted, and an element x of X_i, which is currently under consideration. For this concept x we already know that $c \leq x$, and *top-search* looks at its direct successors with respect to $\prec_i$. Initially, the procedure is called with $x = \top$. For each direct successor y of x we have to check whether it subsumes c. This is done in the procedure *simple-top-subs?*. Since our hierarchy need not be a tree, y may already have been checked before, in which case we have memorized the result of the test, and thus need not invoke the expensive subsumption procedure *subs?*. The direct successors for which the test was positive are collected in a list *Pos-Succ*. If this list remains empty, x is added to the result list; otherwise *top-search* is called for each positive successor, but only if this concept has not been visited before along another path.

The bottom search can be done again in the dual way. It is interesting to note that this top search is in principle the same as the one described by Lipkis [Lipkis, 1982], who implemented the first classification algorithm for KL-ONE. The bottom search described by Lipkis, however, is more efficient than the one given here.

3.3 The Enhanced Traversal Method

Although the simple traversal method is a big advantage compared with the brute force method (see Figure 5 (a)), it still does not exploit all possible information. First, during the top search phase, we can take advantage of tests that have already been performed. Second, in the bottom search phase, we can use the information gained during the top search as well.

Of course, a dual strategy is also possible, i.e., performing the bottom search before the top search and exploiting the information gathered during the bottom search phase. Analyzing Figure 5, it becomes quickly obvious that this strategy would be less efficient, however. In fact, for the simple traversal method—where the top and bottom phase are done in a symmetric way—the top search phase turns out to be a lot faster. Thus it is better to start with this phase because the information gained thereby can then be used to speed up the slower bottom search phase.

When trying to take advantage of tests that have already been performed during top search one can either concentrate on negative information (i.e., that a subsumption test did not succeed) or on positive information (i.e., that a subsumption test was successful).

To *use negative information* during the top search phase one has to check whether for some predecessor z of y the test $c \leq z$ has failed. In this case, we can conclude that $c \not\leq y$ without performing the expensive subsumption test [MacGregor, 1988]. In order to gain maximum advantage, all direct predecessors of y should have been tested before the test is performed on

y [Levinson, 1984]. This can be achieved by using a modified breadth-first search where the already computed hierarchy is traversed in topological order, as described by Ellis [1991] and Levinson [1992]. Alternatively, one can make a recursive call whenever there is a direct predecessor that has not yet been tested. This is what the procedure *enhanced-top-subs?* described in Figure 2 does. If y is not yet marked, the procedure *enhanced-top-subs?* is recursively called for all direct predecessors z of y. As soon as one of these calls returns *false*, one goes to the "else" branch, and marks y "negative." Only if all calls return *true*, the subsumption test *subs?*(y,c) is performed to decide whether y has to be marked "positive" or "negative." If we replace the call of *simple-top-subs?* in *top-search* by a call of *enhanced-top-subs?*, we get the top search part of the enhanced traversal method.

```
enhanced-top-subs?(y,c) =
  if marked?(y,"positive")
    then return true
    elsif marked?(y,"negative")
        then return false
        elsif for all z with y ≺_i z
                   enhanced-top-subs?(z,c)
              and  subs?(y,c)
              then mark(y,"positive")
                   return true
              else mark(y,"negative")
                   return false
        fi
    fi
  fi
```

Figure 2: Top search phase of the enhanced traversal method. The procedure *top-search* is adopted from the simple traversal method, but instead of *simple-top-subs?* it calls *enhanced-top-subs?*

The enhanced top search procedure just described makes maximum use of failed tests. Alternatively, it is possible to *use positive information.* Before checking $c \leq y$, one can look for successors z of y that have passed the test $c \leq z$ [MacGregor, 1988]. If there exists such a successor, one can conclude that $c \leq y$ without performing an actual subsumption test. Although we are only interested in minimizing the number of comparison operations, it should be noted that instead of searching for a successor that has passed the test it is more efficient to propagate positive information up through the subsumption hierarchy. This can be achieved by an easy modification of the procedure *simple-top-subs?*. When the call *subs?*(y,c) yields *true*, not only y is marked "positive," but so are all of y's predecessors. Obviously, this technique cannot be combined with the enhanced top search described in Figure 2 since it reduces the number of subsumption tests only if there are predecessors which have not yet been tested, and enhanced top search tests all predecessors before making a subsumption test.

Neither of these alternatives is uniformly better than the other one, which can be seen by considering the examples described in Figure 3 and 4.

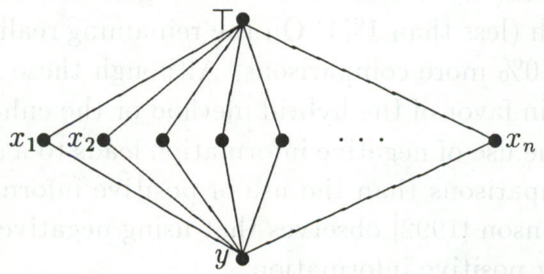


Figure 3: The new element c is a direct successor of y

In the first example, the top-search using negative information makes $n+1$ tests: it first tests x_1, then goes to y, but before testing it, it tests all its direct predecessors, i.e., $x_2, \ldots, x_n$. The top search using positive information makes two tests: first x_1 and then y; the positive result of this second test is propagated to $x_2, \ldots, x_n$.

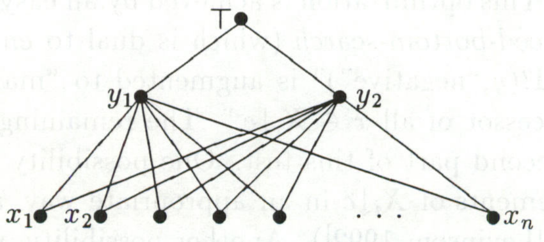


Figure 4: The new element c is a direct successor of y_1, but not a successor of y_2, $x_1, \ldots, x_n$

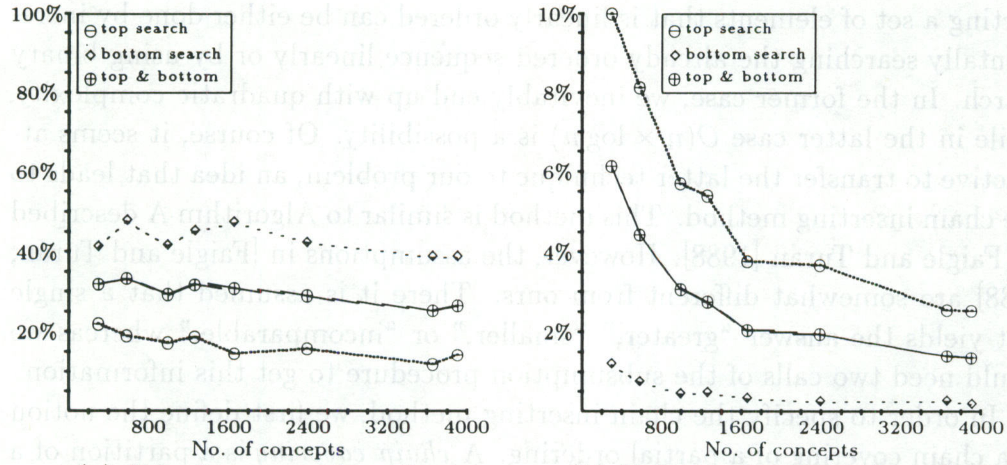
In the second example, the top search using negative information needs only two tests: first it tests y_1, then goes to x_1, but before testing x_1 its direct predecessor y_2 is tested. The negative result of this test prevents $x_1, \ldots, x_n$ from being tested. The top search using positive information tests $n+2$ nodes: first y_1, then all its successors $x_1, \ldots, x_n$, and finally y_2.

However, we have observed significant performance differences for the two different top search strategies. For the random knowledge bases, the method using positive information was only slightly better than the simple traversal method (less than 5%). For this reason, we have also considered a "hybrid method" which propagates positive information up, and negative information down the hierarchy (but does not test all predecessors before testing a node). Propagating negative information down is again achieved by an easy modification of *simple-top-subs?*. When the call of *subs?*(y,c) yields *false*, not only y is marked "negative," but all of y's successors. The hybrid method turned out to be a lot better than just propagating positive information, but it still needed slightly more tests (approx. 5%–10%) than the enhanced top search for all but one of the random knowledge bases. On five of the six realistic knowledge bases the hybrid method was insignificantly faster than the enhanced top search (less than 1%). On the remaining realistic KB, the hybrid method needed 10% more comparisons. Although these results do not seem to be conclusive in favor of the hybrid method or the enhanced top search, it is obvious that the use of negative information leads to a significantly greater reduction of comparisons than the use of positive information. For conceptual graphs, Levinson [1992] observes that using negative information is also better than using positive information.

Now we turn to the *bottom search* phase of the enhanced traversal method. Of course, optimizations dual to the ones described for the top search can be employed here. In addition, the set $X_i{\downarrow}c$ can be used to severely cut down the number of comparisons in the bottom search phase. As mentioned by Lipkis [1982], the search for immediate successors of c can be restricted to the set of successors of $X_i{\downarrow}c$. In fact, the set of candidates for $X_i{\uparrow}c$ is even more constrained. Only elements that are successors of *all* $x \in X_i{\downarrow}c$ can be immediate successors of c [Levinson, 1984; Ellis, 1991; Levinson, 1992]. This optimization is achieved by an easy modification of the procedure *enhanced-bottom-search* (which is dual to *enhanced-top-search*): the test "marked?(y,"negative")" is augmented to "marked?(y,"negative") or y is not a successor of all $x \in X_i{\downarrow}c$." The remaining problem is how to implement the second part of this test. One possibility is to mark the successors of the elements of $X_i{\downarrow}c$ in an appropriate way, and then test these labels (see, e.g., [Levinson, 1992]). Another possibility, which we have used in our tests, is to equip each concept in X_i with a list of all its predecessors in X_i, and test whether $X_i{\downarrow}c$ is contained in the list of predecessors of y.

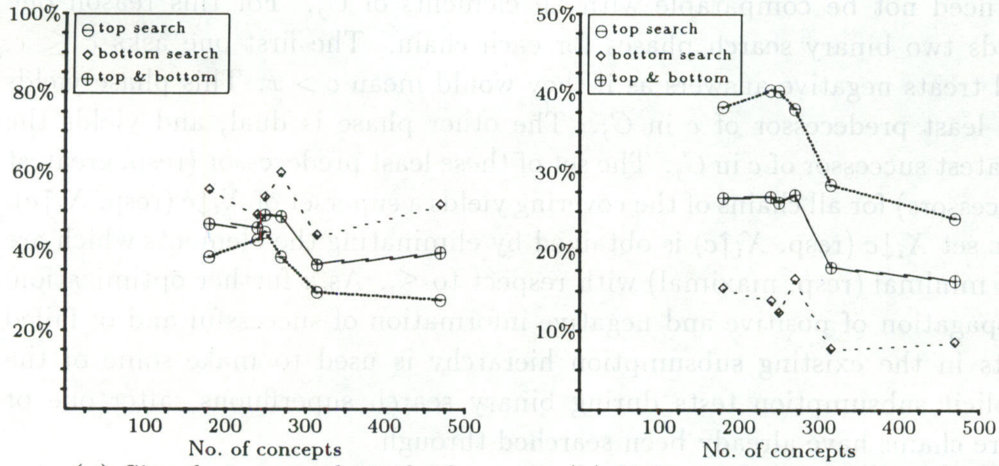
As a result of this optimization, the number of necessary comparison operations can be cut down to a fraction compared with the simple bottom search strategy. Interestingly, we observed a further reduction of comparison operations in case of the real knowledge bases when searching top-down starting at $X_i{\downarrow}c$ instead of searching bottom-up. For the random knowledge bases, no such difference was observed, however. The bottom search described by Ellis [1991] and Levinson [1992] is also done top-down.

The effects of the simple and enhanced traversal method for the random knowledge bases and the realistic knowledge bases as test data are displayed



(a) Simple traversal method (b) Enhanced traversal method

Figure 5: Number of comparison operations relative to *brute force* method for random KBs



(a) Simple traversal method (b) Enhanced traversal method

Figure 6: Number of comparison operations relative to *brute force* method for realistic KBs

in Figures 5 and 6. These graphs present the number of necessary comparisons *relative* to the brute force method for the top search and the bottom search phase, as well as for the entire classification process.

3.4 The Chain Inserting Method

Sorting a set of elements that is linearly ordered can be either done by incrementally searching the already ordered sequence linearly or by using binary search. In the former case, we inevitably end up with quadratic complexity, while in the latter case $O(n \times \log n)$ is a possibility. Of course, it seems attractive to transfer the latter technique to our problem, an idea that leads to the chain inserting method. This method is similar to Algorithm A described by Faigle and Turán [1988]. However, the assumptions in [Faigle and Turán, 1988] are somewhat different from ours. There it is assumed that a single test yields the answer "greater," "smaller," or "incomparable," whereas we would need two calls of the subsumption procedure to get this information.

In order to specify the chain inserting method, we first define the notion of a chain covering of a partial ordering. A *chain covering* is a partition of a partial ordering into *chains*, i.e., totally ordered subsets. Provided we have a chain covering of the set X_i, it is possible to identify the sets $X_i{\downarrow}c$ and $X_i{\uparrow}c$ by binary search in all chains. For a given chain C_j of the covering $X_i = C_1 \cup \cdots \cup C_m$, binary search is used to find the least predecessor and the greatest successor of c in C_j. Since the underlying ordering $\leq$ is only a partial ordering on X, the new element c to be inserted into the chain C_j need not be comparable with all elements of C_j. For this reason one needs two binary search phases for each chain. The first one asks $c \leq x$, and treats negative answers as if they would mean $c > x$. This phase yields the least predecessor of c in C_j. The other phase is dual, and yields the greatest successor of c in C_j. The set of these least predecessor (resp. greatest successors) for all chains of the covering yields a superset of $X_i{\downarrow}c$ (resp. $X_i{\uparrow}c$). The set $X_i{\downarrow}c$ (resp. $X_i{\uparrow}c$) is obtained by eliminating the elements which are not minimal (resp. maximal) with respect to $\leq_i$. As a further optimization, propagation of positive and negative information of successful and of failed tests in the existing subsumption hierarchy is used to make some of the explicit subsumption tests during binary search superfluous, after one or more chains have already been searched through.

We have also considered a "hybrid" method that employs chain inserting for long chains and enhanced traversal afterwards. The idea here is that by binary search in long chains one gets rather quickly into the "center" of the partial ordering, from which propagation of positive and negative information should have the greatest effect.

It is, of course, advisable to use chain coverings with a minimal number of chains. Unfortunately, the computation of minimal chain coverings is non-trivial and takes more than quadratic time [Jungnickel, 1990]. Nevertheless, *simple heuristics* permit the incremental construction of chain coverings that are almost optimal. The heuristic we have used to update the chain covering when a new element c is inserted proceeds as follows. After the sets $X_i{\downarrow}c$

and $X_i{\uparrow}c$ have been computed, c is inserted in the longest chain satisfying one of the following conditions:

1. Binary search has yielded both a least predecessor and greatest successor in the chain, and they are successive elements of the chain. In this case, c is inserted between these two elements in the chain.

2. Binary search has yielded a least predecessor (or greatest successor) in the chain, and it is the least (resp. greatest) element of the chain. In this case, c is inserted below (resp. above) this element in the chain.

If there is no chain satisfying one of these conditions, a new chain consisting of c is created. In our experiments, the chain coverings obtained this way were less than 10% suboptimal.

Some of our empirical results concerning the performance of the chain inserting method are given in Table 2. We have only displayed the results for top search, since the bottom search is almost identical if the optimizations from the enhanced traversal are included.

In addition to the size of the partial order (first column) and the relative number of comparison operations with respect to the enhanced traversal method (last column), also some structural parameters of the partial orders are given. The second column gives the average number of immediate predecessors and successors (where the top and bottom elements are not counted). The third column gives the average number of successors and predecessors, and the fourth and fifth column specify the breadth and depth (including top and bottom), respectively, of the partial order. To be more precise, the entries in the fourth and fifth column are only approximations of the actual breadth and depth. They are the number of chains and the length of the longest chain in the chain covering generated by our heuristic. Taking these numbers in place of the exact numbers is reasonable since our chain insertion method uses the chain coverings generated by the heuristic.

The first group of results was obtained by applying the chain-inserting method to the realistic KBs, the second group gives the results for the random KBs, and the third group specifies the result for the randomly generated partial orders.

To our surprise, the chain inserting method turned out to be not significantly better than the enhanced traversal method. To the contrary, on the realistic KBs it is usually less efficient, except for one case, and the same holds for the random KBs. The "hybrid" version using chain inserting for long chains and enhanced traversal afterwards was also not much better than the pure chain-inserting method. On the other hand, for tests on randomly generated partial orders the chain inserting method in some cases showed a much better performance than the enhanced traversal method. A reason for this behavior could be that, compared to the realistic knowledge bases

No. of nodes	Average degree	Average no. of pred. & succ.	Breadth	Depth	Relative no. of comparisons
184	1.71	5.67	105	6	103.7%
241	1.91	6.38	124	6	100.3%
254	1.99	13.02	135	6	91.8%
269	1.72	5.16	164	7	107.9%
330	1.85	8.13	141	12	110.0%
298	2.36	8.88	142	8	115.7%
583	2.58	12.24	330	7	114.5%
992	2.73	16.77	478	10	111.7%
1263	3.18	16.61	661	11	108.9%
1659	3.19	18.86	927	10	110.3%
2389	3.50	25.49	1188	10	111.3%
3658	3.82	27.20	1703	8	105.3%
3905	4.04	33.95	1858	11	99.9%
301	7.67	42.11	88	8	73.2%
301	8.01	20.69	136	5	100.5%
301	6.40	10.43	168	6	102.7%
301	4.22	5.68	205	4	101.3%
586	9.93	72.55	144	9	67.2%
586	12.08	38.79	224	7	96.3%
586	10.39	20.42	301	7	103.3%
586	7.72	11.50	353	5	102.9%
995	5.52	250.24	85	28	16.2%
995	12.46	125.94	226	11	51.8%
995	16.40	62.88	354	9	91.4%
995	13.58	28.38	506	6	105.1%
1266	5.78	321.19	100	30	12.9%
1266	13.82	169.95	259	13	44.2%
1266	18.22	76.87	438	9	89.8%
1266	17.82	41.70	592	6	101.1%

Table 2: Number of comparison operations in top search of the chain-inserting method *relative* to enhanced traversal. The first group gives results for the realistic KBs, the second group for the random KBs, and the third group for the randomly generated partial orders.

we used in our tests, the randomly generated partial orders have a much higher connectivity (which means that propagation of positive and negative information has more effect) and permit longer chains (which makes binary search more important). In fact, the parameters *average number of predecessors and successors* and *breadth* seem to be relevant for the efficiency of the chain-inserting method. If the first parameter is high and the second is low relative to the size of the partial order, then the chain-inserting method performs much better than enhanced traversal.

The chain inserting method may thus become more interesting for knowledge bases defining relatively deep hierarchies with high connectivity. Additionally, it seems possible to exploit the chain covering in order to implement storage compression techniques as described by Jagadish [1989]. Finally, it should be noted that the overhead of the chain-inserting method is not significantly higher than the overhead of the enhanced traversal method. In fact, in our implementation the chain-inserting method required slightly less overhead than the enhanced traversal method.

4 Obvious Subsumption Relationships

In this section we describe some further techniques for avoiding subsumption tests by exploiting relations which are obvious when looking at the syntactic structure of concept definitions.[4] These pre-tests require only little effort but can speed up the classification process significantly. We consider three different optimizations which apply to different stages of the classification process.

The first technique can be used prior to the top search. It applies when the description of the concept c that we want to insert is conjunctive (which is the case for the majority of concepts, in particular if we consider the existing real knowledge bases). If this description mentions x explicitly as a conjunct, then it is obviously the case that $c \leq x$. We call such concepts x *told subsumers* of c. Of course, if x is also a conjunctively defined concept, it may have told subsumers as well, and these (and their told subsumers, etc.) can be included into the list of told subsumers of c. It is rather easy to compile this list while reading in the concept definitions. The information that c is subsumed by its told subsumers can be propagated through the existing hierarchy $(X_i, \prec_i)$ prior to the top search, e.g., by pre-setting the markers used in the traversal method to "positive" for the told subsumers and all their predecessors. A prerequisite for this optimization technique to be effective is that the told subsumers of c are already contained in X_i. This can be achieved by inserting concepts following the so-called "definition-order." This order can be formally defined as follows: We say that a concept x directly uses a concept y iff y occurs in the definition of x. Let "uses" be the transitive closure of "directly uses." Then x comes in the definition-order after y if x uses y.

[4]These techniques are probably used in all systems, see, e.g. [Peltason *et al.*, 1989].

Assuming that concepts are inserted in the subsumption hierarchy following the definition-order, another optimization can be applied. The bottom search phase can be completely avoided if a *primitive concept* (i.e., a concept that is described by giving only necessary conditions) has to be classified. In fact, such a concept c can only subsume the bottom concept and concepts whose definitions use c. Since the second type of possible subsumees is not yet present in the actual hierarchy when inserting along the definition-order, the result of the bottom search is just the bottom concept $\perp$. Considering the fact that in realistic KBs the majority of concepts (60%-90%) are primitive, this optimization can save most of the subsumption calls during the bottom search phase. Combining the two optimization techniques led to a saving of 10% to 20% with respect to the pure enhanced traversal method for the realistic knowledge bases. In case of the random knowledge bases, the savings where even greater, as can be seen from Figure 7.

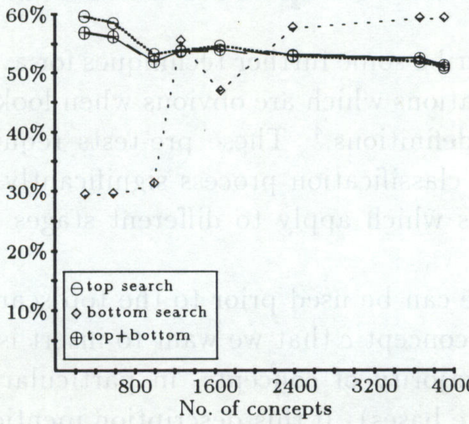


Figure 7: Number of necessary comparisons when exploiting obvious subsumption relations relative to pure *enhanced traversal* method for random KBs

A final optimization technique can be used as a pre-test before calling the subsumption algorithm. It makes use of the fact that a concept c is subsumed by a *primitive concept* x if, and only if, the completely expanded form of c contains the "primitive component of x" (see [Nebel, 1990a, p. 54-56]) as a conjunct. By extracting and caching the "primitive components" of all concepts, it becomes possible to check whether a subsumption relation is possible by comparing the sets of primitive components. If this test gives a negative result, the subsumption algorithm need not be called. Although such a test overlaps with computations the subsumption algorithm does, it is much faster than the subsumption test. For this reason, this pre-test pays off if most of the subsumption calls can be avoided, which was indeed the case for our test data. Our experiments indicate that the number of calls of the subsumption algorithm can be again reduced by 50%-60%, if this technique is applied.

5 The Subsumption Algorithm

In this section we consider two possible optimizations of the subsumption algorithm, and describe the effects they have on the performance of classification for our test knowledge bases. Let us first reconsider the two types of subsumption algorithms usually implemented in terminological systems.

In almost all terminological representation systems other than KRIS *structural subsumption algorithms* are employed (e.g. CLASSIC, LOOM, BACK). Such algorithms basically proceed as follows. First, the concepts are normalized, i.e., they are transformed into equivalent normal forms. Subsumption between normalized concepts is a kind of structural comparison where each subexpression of the first concept must have a counterpart in the other concept. This algorithmic technique allows one to develop efficient subsumption algorithms which are easily shown to be sound. However, for expressive terminological languages these algorithms are usually not complete, and it is not clear how the technique could be extended in order to build complete structural subsumption algorithms.

Using a different paradigm, in the past years sound and *complete* subsumption algorithms for a large class of terminological languages have been developed (e.g. [Schmidt-Schauß and Smolka, 1991; Hollunder *et al.*, 1990]). Most of these algorithms are designed as *satisfiability checking algorithms*. These algorithms are model generation procedures, and are similar to first-order tableaux calculus, with the main difference that the specific structure of concept descriptions allows one to impose an appropriate control that ensures termination. Since a concept A subsumes a concept B if, and only if, $\neg A \sqcap B$ is not satisfiable, i.e., there does not exist an interpretation which interprets $\neg A \sqcap B$ as a non-empty set, a satisfiability algorithm in fact can be used to solve the subsumption problem. In order to check whether a given concept C is satisfiable, the tableaux-based algorithm tries to generate a finite interpretation in which C is interpreted as a non-empty set. This generation process is complete in the sense that if it fails, i.e., an obvious contradiction occurs, we can conclude that C is not satisfiable; otherwise C is satisfiable. An obvious contradiction in the model generating process occurs, for example, if some element is constrained to be both instance of a "primitive component" and its complement—which is impossible.

It is well-known that subsumption of concepts defined in a cycle-free terminology can be reduced easily to subsumption of concept terms which do not refer to other concept definitions of the terminology (so-called *expanded concept terms*) [Nebel, 1990a]. For conceptual simplicity both types of subsumption algorithms are usually described in the literature as taking expanded concept terms as arguments, which precludes the exploitation of previously computed subsumption relationships.

5.1 The Optimizations

However, almost all terminological representation systems take advantage of previously computed subsumption relationships. To illustrate how this can be done for a structural subsumption algorithm, suppose that C and D are normalized concept descriptions. As mentioned above, structural subsumption between C and D means to find for each subexpression C' of C a corresponding subexpression D' of D. Often, in turn, these subexpressions have to be tested for subsumption. In case C' and D' are concept names of possibly defined concepts, and we already know a subsumption relationship between C' and D', it is not necessary to call the subsumption algorithm recursively for (the expanded form of) C' and D'. Thus, it is rather natural and straightforward to incorporate the use of already computed subsumption relations into a structural subsumption algorithm. It should be noted that it is an essential requirement *not* to completely expand the concept definitions before checking subsumption since otherwise the concept names for which subsumption relationships are already known would be lost. Further, it is necessary to classify the concepts according to the "definition-order" mentioned in the previous section.

In contrast to other terminological systems, KRIS employs a satisfiability algorithm to determine subsumption relationships between concepts. Since a satisfiability algorithm does not recursively call subsumption algorithms but satisfiability algorithms, it is not obvious how to exploit previously computed subsumption relationships. A closer look, however, reveals that a satisfiability algorithm may detect a contradiction earlier during model generation if previously computed subsumption relationships are taken into account. To see this, suppose that we already know that a defined concept A subsumes a defined concept B. If during the model generation an element is constrained to be both instance of $\neg A$ and B, a contradiction can be detected without expanding the definitions of A and B. Again, this approach only works if the concept definitions are not expanded before starting to check satisfiability.

If expansion is done "by need" during the satisfiability test, one has to decide in which order to expand the concept names. It is easy to see that this order may have considerable impact on the runtime behavior. For example, assume that we are testing $A \sqcap B$ for satisfiability where in the TBox A is defined by a very large concept description and B is defined to be $\neg A \sqcap C$. If B is expanded first, the contradiction between A and $\neg A$ is detected at once. On the other hand, if A is expanded first, detecting the contradiction between the large descriptions associated with A and its negation may be rather time-consuming, depending on the structure of the description.

One way of avoiding this problem is to expand concept names according to the inverse of their definition-order, which in the above example would mean that we expand B before A, because the definition of B refers to A. Of course, this means that for each expansion operation one has to go through the list of all expandable names, and look for a maximal one with respect to the definition-order. For our tests we have used another solution, which avoids searching for a maximal name, but may use more space. Here one expands in arbitrary order, but when a name is expanded it is not removed, but just marked as expanded. If, in our example, A is expanded before B, we then still have the name A, and as soon as B is expanded it yields the contradiction with $\neg A$.

In order to gain experience in how to optimize the satisfiability algorithm to be employed in KRIS, we implemented the following three versions.

1. The first one takes *completely expanded* concept descriptions as input. Since these descriptions do not contain names of defined concepts, obvious contradictions can only be detected between "primitive components," i.e., concept names which are not defined in the TBox.

2. The second one *successively expands* the concept descriptions during model generation, but keeps the names, as described above. This allows the algorithm to detect obvious contradictions not only between primitive components but also between names of defined concepts.

3. The third version is a refinement of the second one in that already computed subsumption relationships are taken into account when looking for obvious contradictions.

5.2 Empirical Results and Analysis

It turns out that the first version is significantly slower than the second one, a result we did expect. The main reason for this behavior is that the number of recursive calls of the satisfiability algorithm is reduced due to obvious contradictions detected between names of defined concepts. As a consequence, the runtime of the second version is reduced by 40-60% relative to the first version (see Figure 8, which displays the results for the random knowledge bases).

A result we did *not* expect is that the behavior of the third version is no better than of the second, which means that trying to exploit already computed subsumption relationships does not pay off. The reason for this behavior seems to be that—at least for the test data—only a few contradictions are detected by using already computed subsumption relationships.

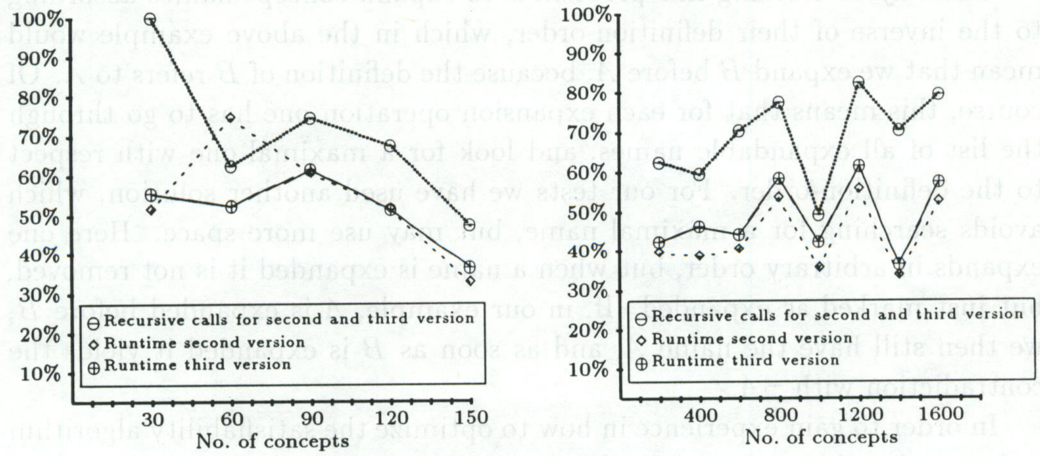


Figure 8: Runtime and number of recursive calls of the second and third version's satisfiability algorithm relative to the algorithm taking completely expanded concept terms as input (first version) for random KBs

This is indicated by the fact that the number of recursive calls of the satisfiability algorithm does not significantly decrease when going from the second to the third version. However, the test of whether a set of negated and unnegated concept names is contradictory w.r.t. already computed subsumption relationships is more complex than just searching for complementary names, which explains that the third version's runtime behavior is even slightly worse than the second one's (see Figure 8).

This result is all the more surprising since using computed subsumption relationships during classification is an optimization technique employed by most terminological systems. The reason why it may pay off for other systems could be that these systems first normalize, and during this normalization phase auxiliary concepts may be introduced. For example, assume that C is defined by the description $\forall R.A \sqcap \forall R.B$, and D by $\forall R.A$. The normalization procedure may introduce a new concept name E, define it as $A \sqcap B$, and modify the definition of C to $\forall R.E$. Now the subsumption relationship between A and the auxiliary concept E—which is found first if the terminology is classified according to the definition-order—immediately entails that D subsumes C. Thus classification of the terminology with the auxiliary concepts allows one to exploit previously computed subsumption relationships more often. On the other hand, it has the disadvantage that in general a lot more concepts have to be classified.

Another interesting behavior we observed is due to the interaction between different optimization techniques. The optimizations described in the previous two sections try to avoid subsumption tests, whereas the present section is concerned with speeding up the subsumption test. Ideally, one could expect that these optimizations are independent. This means that the overall speedup factor is the product of the speedup factors of the individual optimizations. This can only be true if the optimizations apply uniformly to all situations, however.

If the optimizations apply to special cases only, subsumption avoidance optimizations and subsumption test optimization may aim at similar special cases and lead to the situation that subsumption tests are avoided which have neglectable computational costs in any case.

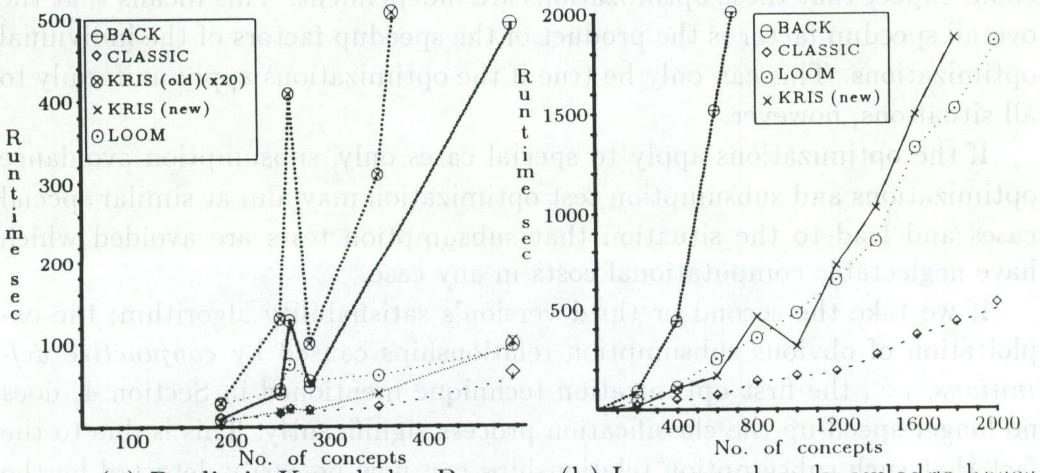
If we take the second or third version's satisfiability algorithm, the exploitation of obvious subsumption relationships caused by *conjunctive definitions,* i.e., the first optimization technique mentioned in Section 4, does no longer speed up the classification process significantly. This is due to the fact that such subsumption relationships can now be easily detected by the satisfiability algorithms. For example, let C be a concept that is defined to be the conjunction of $C_1, \ldots, C_m$, where the C_i are defined concepts as well. The obvious subsumption relationship between C_i and C is immediately detected by the second and third version of the satisfiability algorithm, due to an obvious contradiction between C_i and $\neg C_i$.

6 Conclusion

We have described and analyzed different optimization techniques for the classification process in terminological representation systems. Interestingly, two of the most promising techniques, namely, the chain inserting method for computing the representation of a partial order and the exploitation of already computed subsumption relations in the subsumption algorithm, did not lead to the expected performance increase in case of realistic knowledge bases.

A further interesting result is that the optimization technique on the first level, which we called *enhanced traversal method* and which turned out to be the most promising method in our case, also gives good results for managing hierarchies of conceptual graphs [Levinson, 1984; Ellis, 1991; Ellis and Levinson, 1992], indicating that these hierarchies are probably structurally similar to those induced by terminological knowledge bases.

As a result of our empirical analysis, the optimization techniques that came off best were incorporated in the KRIS system. These, together with more conventional optimizations on the implementation level, led to a significant speed up. Whereas the unoptimized version was orders of magnitude slower than the fastest system tested in [Heinsohn *et al.*, 1992], the new version has now a runtime behavior similar to that of the other systems on the test data used there.



(a) Realistic Knowledge Bases (b) Large random knowledge bases

Figure 9: Runtime performance for realistic and large random knowledge bases

Figure 9(a) displays the runtime of the new KRIS version for the realistic knowledge bases and contrasts them with the runtime figures given in [Heinsohn *et al.*, 1992]. Figure 9(b) gives the results for large random knowledge bases.[5]

It should be noted, however, that all the knowledge bases used in the test are formulated using quite limited terminological languages. An interesting open problem is the development of further optimization techniques for more powerful terminological languages containing also disjunction and negation and of specific optimization techniques for assertional reasoning.

Acknowledgements

We would like to thank Uwe Utsch for implementing the different subsumption strategies, Hans-Jürgen Bürckert, Jochen Heinsohn, Armin Laux, and Werner Nutt for helpful discussions concerning the topics described in this paper, and Alex Borgida and Peter Patel-Schneider for helpful comments on an earlier version of this paper.

This work has been supported by the German Ministry for Research and Technology (BMFT) under research contracts ITW 8901 8 and ITW 8903 0 and by the Italian National Research Council (CNR), project "Sistemi Informatici e Calcolo Parallelo."

[5]The description of the runtime behavior of the systems in [Heinsohn *et al.*, 1992] refers to system versions as of 1990 and does not necessarily reflect the performance of more recent versions.

References

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

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

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

[Brachman *et al.*, 1985] Ronald J. Brachman, Victoria Pigman Gilbert, and Hector J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts in KRYPTON. In IJCAI-85 [1985], pages 532–539.

[Brachman, 1977] Ronald J. Brachman. *A Structural Paradigm for Representing Knowledge*. PhD thesis, Havard University, 1977.

[Cattoni and Franconi, 1990] Roldano Cattoni and Enrico 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, October 1990. North-Holland.

[Donini *et al.*, 1991a] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner 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, April 1991. Morgan Kaufmann.

[Donini *et al.*, 1991b] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 458–465, Sydney, Australia, August 1991. Morgan Kaufmann.

[Edelmann and Owsnicki, 1986] Jürgen Edelmann and Bernd Owsnicki. Data models in knowledge representation systems: A case study. In C.-R. Rollinger and W. Horn, editors, *GWAI-86 und 2. Österreichische Artificial-Intelligence-Tagung*, pages 69–74, Ottenstein, Austria, September 1986. Springer-Verlag.

[Ellis, 1991] Gerard Ellis. Compiled hierarchical retrieval. In *Proceedings of the 6th Annual Conceptual Graphs Workshop*, 1992.

[Ellis and Levinson, 1992] Gerard Ellis and Robert Levinson. The birth of PEIRCE: A conceptual graphs workbench. In H. Pfeiffer, editor, *Proceedings of the Seventh Annual Conceptual Graphs Workshop*, Las Cruces, New Mexico, July 8-10, 1992.

[Faigle and Turán, 1988] U. Faigle and Gy. Turán. Sorting and recognition problems for ordered sets. *SIAM J. Computing*, 17(1):100–113, 1988.

[Heinsohn *et al.*, 1992] Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich. An empirical analysis of terminological representation systems. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 767–773, San Jose, CA, July 1992. MIT Press.

[Hollunder *et al.*, 1990] Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In L. C. Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 348–353, Stockholm, Sweden, August 1990. Pitman.

[IJCAI-85, 1985] *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985.

[Jagadish, 1989] H. V. Jagadish. A compressed transitive closure technique for efficient fixed-point query processing. In L. Kerschberg, editor, *Expert Database Systems—Proceedings From the 2nd International Conference*, pages 423–446, Menlo Park, CA, 1989. Benjamin/Cummings.

[Jungnickel, 1990] Dieter Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI Wissenschaftsverlag, Mannheim, Germany, 2nd edition, 1990.

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

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

[Levinson, 1984] Robert Levinson. A self-organizing retrieval system for graphs. In *Proceedings of the 3rd National Conference of the American Association for Artificial Intelligence*, pages 203–206, Austin, Texas, 1984.

[Levinson, 1992] Robert Levinson. Pattern associativity and the retrieval of semantic networks. *Journal of Computers & Mathematics with Applications*, 23(6–9):573–600, 1992.

[Lipkis, 1982] Thomas Lipkis. A KL-ONE classifier. In J. G. Schmolze and R. J. Brachman, editors, *Proceedings of the 1981 KL-ONE Workshop*, pages 128–145, Cambridge, MA, 1982. The proceedings have been published as BBN Report No. 4842 and Fairchild Technical Report No. 618.

[MacGregor, 1988] Robert MacGregor. A deductive pattern matcher. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 403–408, Saint Paul, MI, August 1988.

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

[Mays *et al.*, 1991] Eric Mays, Robert Dionne, and Robert Weida. K-Rep system overview. *SIGART Bulletin*, 2(3):93–97, June 1991.

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

[Nebel, 1990a] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1990.

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

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

[Patel-Schneider, 1984] Peter F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pages 11–16, Denver, Colo., 1984. An extended version including a KANDOR system description is available as AI Technical Report No. 37, Palo Alto, CA, Schlumberger Palo Alto Research, October 1984.

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

[Peltason *et al.*, 1989] Christof Peltason, Albrecht Schmiedel, Carsten Kindermann, and Joachim Quantz. The BACK system revisited. KIT Report 75, Department of Computer Science, Technische Universität Berlin, Berlin, Germany, September 1989.

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

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

[Schmidt-Schauß, 1989] Manfred 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.

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

[Vilain, 1985] Marc B. Vilain. The restricted language architecture of a hybrid representation system. In IJCAI-85 [1985], pages 547–551.

[Winkler, 1985] P. Winkler. Random orders. *Order*, 1:317–331, 1985.

[Woods, 1991] William A. Woods. Understanding subsumption and taxonomy: A framework for progress. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 45–94. Morgan Kaufmann, San Mateo, CA, 1991.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-92-10
M. Bauer: An Interval-based Temporal Logic in a Multivalued Setting
17 pages

RR-92-11
Susane Biundo, Dietmar Dengler, Jana Koehler: Deductive Planning and Plan Reuse in a Command Language Environment
13 pages

RR-92-13
Markus A. Thies, Frank Berger: Planbasierte graphische Hilfe in objektorientierten Benutzungsoberflächen
13 Seiten

RR-92-14
Intelligent User Support in Graphical User Interfaces:
1. InCome: A System to Navigate through Interactions and Plans
   Thomas Fehrle, Markus A. Thies
2. Plan-Based Graphical Help in Object-Oriented User Interfaces
   Markus A. Thies, Frank Berger

22 pages

RR-92-15
Winfried Graf: Constraint-Based Graphical Layout of Multimodal Presentations
23 pages

RR-92-16
Jochen Heinsohn, Daniel Kudenko, Berhard Nebel, Hans-Jürgen Profitlich: An Empirical Analysis of Terminological Representation Systems
38 pages

RR-92-17
Hassan Aït-Kaci, Andreas Podelski, Gert Smolka: A Feature-based Constraint System for Logic Programming with Entailment
23 pages

RR-92-18
John Nerbonne: Constraint-Based Semantics
21 pages

RR-92-19
Ralf Legleitner, Ansgar Bernardi, Christoph Klauck: PIM: Planning In Manufacturing using Skeletal Plans and Features
17 pages

RR-92-20
John Nerbonne: Representing Grammar, Meaning and Knowledge
18 pages

RR-92-21
Jörg-Peter Mohren, Jürgen Müller
Representing Spatial Relations (Part II) -The Geometrical Approach
25 pages

RR-92-22
Jörg Würtz: Unifying Cycles
24 pages

RR-92-23
Gert Smolka, Ralf Treinen:
Records for Logic Programming
38 pages

RR-92-24
Gabriele Schmidt: Knowledge Acquisition from Text in a Complex Domain
20 pages

RR-92-25
Franz Schmalhofer, Ralf Bergmann, Otto Kühn, Gabriele Schmidt: Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations
12 pages

RR-92-26
Franz Schmalhofer, Thomas Reinartz, Bidjan Tschaitschian: Intelligent documentation as a catalyst for developing cooperative knowledge-based systems
16 pages

RR-92-27
Franz Schmalhofer, Jörg Thoben: The model-based construction of a case-oriented expert system
18 pages

RR-92-29
Zhaohui Wu, Ansgar Bernardi, Christoph Klauck: Skeletel Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

RR-92-30
Rolf Backofen, Gert Smolka
A Complete and Recursive Feature Theory
32 pages

RR-92-31
Wolfgang Wahlster
Automatic Design of Multimodal Presentations
17 pages

RR-92-33
Franz Baader: Unification Theory
22 pages

RR-92-34
Philipp Hanschke: Terminological Reasoning and Partial Inductive Definitions
23 pages

RR-92-35
Manfred Meyer:
Using Hierarchical Constraint Satisfaction for Lathe-Tool Selection in a CIM Environment
18 pages

RR-92-36
Franz Baader, Philipp Hanschke:
Extensions of Concept Languages for a Mechanical Engineering Application
15 pages

RR-92-37
Philipp Hanschke: Specifying Role Interaction in Concept Languages
26 pages

RR-92-38
Philipp Hanschke, Manfred Meyer:
An Alternative to Θ-Subsumption Based on Terminological Reasoning
9 pages

RR-92-40
Philipp Hanschke, Knut Hinkelmann: Combining Terminological and Rule-based Reasoning for Abstraction Processes
17 pages

RR-92-41
Andreas Lux: A Multi-Agent Approach towards Group Scheduling
32 pages

RR-92-42
John Nerbonne:
A Feature-Based Syntax/Semantics Interface
19 pages

RR-92-43
Christoph Klauck, Jakob Mauss: A Heuristic driven Parser for Attributed Node Labeled Graph Grammars and its Application to Feature Recognition in CIM
17 pages

RR-92-44
Thomas Rist, Elisabeth André: Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP
15 pages

RR-92-45
Elisabeth André, Thomas Rist: The Design of Illustrated Documents as a Planning Task
21 pages

RR-92-46
Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, Wolfgang Wahlster: WIP: The Automatic Synthesis of Multimodal Presentations
19 pages

RR-92-47
Frank Bomarius: A Multi-Agent Approach towards Modeling Urban Traffic Scenarios
24 pages

RR-92-48
Bernhard Nebel, Jana Koehler:
Plan Modifications versus Plan Generation: A Complexity-Theoretic Perspective
15 pages

RR-92-49
Christoph Klauck, Ralf Legleitner, Ansgar Bernardi:
Heuristic Classification for Automated CAPP
15 pages

RR-92-50
Stephan Busemann:
Generierung natürlicher Sprache
61 Seiten

RR-92-51
Hans-Jürgen Bürckert, Werner Nutt:
On Abduction and Answer Generation through Constrained Resolution
20 pages

RR-92-52
Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul: PHI - A Logic-Based Tool for Intelligent Help Systems
14 pages

RR-92-54
Harold Boley: A Direkt Semantic Characterization of RELFUN
30 pages

RR-92-55
John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, Stephan Oepen: Natural Language Semantics and Compiler Technology
17 pages

RR-92-58
Franz Baader, Bernhard Hollunder:
How to Prefer More Specific Defaults in Terminological Default Logic
31 pages

RR-92-59
Karl Schlechta and David Makinson: On Principles and Problems of Defeasible Inheritance
14 pages

RR-93-02
Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, Thomas Rist:
Plan-based Integration of Natural Language and Graphics Generation
50 pages

RR-93-03
Franz Baader, Berhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi:
An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 pages

RR-93-05
Franz Baader, Klaus Schulz: Combination Techniques and Decision Problems for Disunification
29 pages

RR-92-60
Karl Schlechta: Defaults, Preorder Semantics and Circumscription
18 pages

DFKI Technical Memos

TM-91-12
Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13
Knut Hinkelmann: Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14
Rainer Bleisinger, Rainer Hoch, Andreas Dengel:
ODA-based modeling for document analysis
14 pages

TM-91-15
Stefan Bussmann: Prototypical Concept Formation
An Alternative Approach to Knowledge Representation
28 pages

TM-92-01
Lijuan Zhang: Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02
Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03
Mona Singh:
A Cognitiv Analysis of Event Structure
21 pages

TM-92-04
Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer:
On the Representation of Temporal Knowledge
61 pages

TM-92-05
Franz Schmalhofer, Christoph Globig, Jörg Thoben:
The refitting of plans by a human expert
10 pages

TM-92-06
Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08
Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27 pages

DFKI Documents

D-92-06
Hans Werner Höper: Systematik zur Beschreibung von Werkstücken in der Terminologie der Featuresprache
392 Seiten

D-92-07
Susanne Biundo, Franz Schmalhofer (Eds.): Proceedings of the DFKI Workshop on Planning
65 pages

D-92-08
Jochen Heinsohn, Bernhard Hollunder (Eds.): DFKI Workshop on Taxonomic Reasoning Proceedings
56 pages

D-92-09
Gernod P. Laufkötter: Implementierungsmöglichkeiten der integrativen Wissensakquisitionsmethode des ARC-TEC-Projektes
86 Seiten

D-92-10
Jakob Mauss: Ein heuristisch gesteuerter Chart-Parser für attributierte Graph-Grammatiken
87 Seiten

D-92-11
Kerstin Becker: Möglichkeiten der Wissensmodellierung für technische Diagnose-Expertensysteme
92 Seiten

D-92-12
Otto Kühn, Franz Schmalhofer, Gabriele Schmidt: Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13
Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-14
Johannes Schwagereit: Integration von Graph-Grammatiken und Taxonomien zur Repräsentation von Features in CIM
98 Seiten

D-92-15
DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16
Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17
Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.): UM92: Third International Workshop on User Modeling, Proceedings
254 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-92-18
Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19
Stefan Dittrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21
Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

D-92-23
Michael Herfert: Parsen und Generieren der Prolog-artigen Syntax von RELFUN
51 Seiten

D-92-24
Jürgen Müller, Donald Steiner (Hrsg.): Kooperierende Agenten
78 Seiten

D-92-25
Martin Buchheit: Klassische Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26
Enno Tolzmann: Realisierung eines Werkzeugauswahlmoduls mit Hilfe des Constraint-Systems CONTAX
28 Seiten

D-92-27
Martin Harm, Knut Hinkelmann, Thomas Labisch: Integrating Top-down and Bottom-up Reasoning in COLAB
40 pages

D-92-28
Klaus-Peter Gores, Rainer Bleisinger: Ein Modell zur Repräsentation von Nachrichtentypen
56 Seiten

An Empirical Analysis of Optimization Techniques for Terminological Representation Systems or: 'Making KRIS get a move on'

RR-93-03
Research Report

Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi