High Performance Reasoning with Very Large Knowledge Bases

Volker Haarslev and Ralf Möller
University of Hamburg, Computer Science Department
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany

Abstract

In this contribution we present an empirical analysis of the performance of the $\mathcal{ALCNH}_{R^+}$ description logic system RACE applied to TBoxes with a very large number of primitive concept definitions. Adaptions of previously known techniques as well as new optimization techniques for efficiently dealing with these kinds of knowledge bases are discussed.

1 Motivation

In application projects it is often necessary to deal with TBoxes with a large number of axioms. In addition, in many applications only a small subset of the axioms are true generalized concept inclusions (GCIs). In most cases, axioms are concept introduction axioms. Usually it has been argued that only systems based on incomplete calculi can deal with knowledge bases with more than 100,000 axioms of this kind. In this contribution we present an empirical analysis of the performance of the $\mathcal{ALCNH}_{R^+}$ description logic system RACE [4, 6] applied to knowledge bases of this size.[1] It is shown that description logic systems based on sound and complete algorithms are particularly useful for simple but large knowledge bases consisting mainly of primitive concept definitions. A knowledge base is called *simple* if no meta constraints remain after the absorption phase [9] and there exist (almost) no defined concepts.

As an example knowledge base we consider a reconstruction of important parts of the UMLS (Unified Medical Language System) [12] using description logic representation techniques. The reconstruction is described in [13] and introduces a specific scheme that uses several concept names to represent subset as well as composition aspects of each concept (word) mentioned in the UMLS metathesaurus. For instance, for the notion of a 'heart', the following axioms

[1]A convenient pronunciation of $\mathcal{ALCNH}_{R^+}$ is ALC-nature.

for heart structures (suffix ‘s’), heart parts (suffix ‘p’) and heart entities (no suffix) are declared (see [13] for details):

$$\begin{aligned}\textbf{ana\_heart} &\sqsubseteq \mathsf{ana\_heart\_s} \sqcap \mathsf{ana\_hollow\_viscus} \sqcap \\ &\quad \mathsf{umls\_body\_part\_organ\_or\_organ\_component} \\ \textbf{ana\_heart\_s} &\sqsubseteq \mathsf{ana\_hollow\_viscus\_s} \sqcap \mathsf{ana\_cardiovascular\_system\_p} \\ \textbf{ana\_heart\_p} &\sqsubseteq \neg\mathsf{ana\_heart} \sqcap \mathsf{ana\_heart\_s} \sqcap \exists_{\geq 1}\ \mathsf{anatomical\_part\_of\_ana\_heart}\end{aligned}$$

Note the disjointness declaration between ana_heart_p and ana_heart. The following role axiom is generated as well.

$$\textbf{anatomical\_part\_of\_ana\_heart} \sqsubseteq \mathsf{anatomical\_part\_of\_ana\_hollow\_viscus}$$

It is beyond the scope of this paper to discuss the pros and cons of specific modeling techniques used in the UMLS reconstruction. In the next section, optimization techniques for efficiently dealing with these kinds of knowledge bases are presented.

2 Optimization Techniques

Modern DL systems such as RACE offer (at least) two operations for TBoxes: classification and coherence checking [7]. Classification is the process of computing the most-specific subsumption relationships (“parents” and “children”) of every concept name with other concept names mentioned in a TBox. Coherence checking determines all concept names which are incoherent.

Our findings indicate that state-of-the-art techniques currently employed for fast classification of TBoxes have to be extended in order to cope with large knowledge bases of the above-mentioned kind. In the following we will give a sketch of techniques that enable RACE to deal with very large knowledge bases.

2.1 Topological Sorting for Achieving Quasi Definition Order

For TBox classification the RACE system employs the marking and propagation techniques introduced in [1]. The parents and children of a certain concept name are computed in so-called ‘top search’ and ‘bottom search’ traversal phases, respectively. For large knowledge bases it is particularly important to avoid as many traversals as possible. Let us assume, a TBox to be classified can be transformed such that no meta constraints but maybe cyclic (primitive) concept definitions exist. Then, if concepts are classified in a so-called ‘definition order’, the bottom search phase can be omitted for concept names for which only a primitive concept definition exists [1]. According to [1] we assume that a concept name A ‘directly uses’ a concept name B if B occurs in the concept on the right-hand side of the definition of A. The relation ‘uses’ is the transitive closure of ‘directly uses’. If A uses B then A comes before B in the definition order. For acyclic TBoxes (i.e. the uses relation is irreflexive) with concept introduction axioms only, the set of concepts can be processed in definition order, i.e. a concept is not classified until all of the concepts used in its definition are classified. In this case the set of children of a concept name consists only of the bottom concept. Thus, a common syntactical restriction for description logic systems is to accept only TBox declarations that do not include so-called forward references. However, for a language such as $\mathcal{ALCNH}_{R^+}$, which offers cyclic axioms and GCIs, in general, the bottom search phase cannot be skipped [9, page 103].

Unfortunately, in the UMLS examples there are many forward references involved in value restrictions and existential restrictions (i.e. modalities). Thus, the definition order of concept names has to be computed in a preprocessing step. In addition, a slightly less strict notion of definition order has been developed. We assume a relation 'directly uses non-modal' similar to 'directly uses' but with references occurring in the scope of quantifiers not considered. Again 'uses non-modal' is the transitive closure of 'directly uses non-modal'. For acyclic concepts the 'uses non-modal' relation induces a partial order relation on concept names. All concept names involved in a cycle are treated as one node (i.e. a set S_i) w.r.t. the partial order. Using a topological sorting algorithm the partial order can be serialized such that a total order between concept names (or sets of concept names) is defined. We call the serialization a "quasi definition order".

During classification of a TBox with RACE the concept names are processed in the order given by the linearization w.r.t. topological sorting. For each primitive concept that is not a member of a set S_i, we claim that the bottom search can be disabled. The 'uses non-modal' relation and the quasi definition order serialization ensures that either all concepts that are potential subconcepts of a certain primitive concept A are inserted after A has been inserted into the subsumption lattice or the bottom search is indeed performed. The quasi definition order is conservative w.r.t. the potential subsumers (note that $\mathcal{ALCNH}_{R^+}$ does not support inverse roles). Moreover, in a basic subsumption test the subsumption lattice under construction is never referred to. Thus, strict definition order classification is not necessary.

Topological sorting is of order $n + e$ where e is the number of given 'uses non-modal' relationships. Thus, we have approximately $O(n \log n)$ steps while the bottom search procedure requires $O(n^2)$ steps in the worst case. Note that in [1] no experiments are discussed that involve the computation of a serialization given a TBox with axioms not already in (strict) definition order.

2.2 Dealing with Domain and Range Restrictions

In order to avoid disjunctions, GCIs for domain restrictions are dealt with by RACE with a generalized kind of lazy unfolding. In a similar way as for names, all situations where unfolding of concept terms $\exists\,\mathsf{R}\,.\,\mathsf{D}$ w.r.t. axioms of the form $\exists\,\mathsf{R}\,.\,\top \sqsubseteq \mathsf{C}$ must occur can be easily identified, i.e. unfolding of a domain restriction for a role R is applied whenever an assertion $\mathsf{i}:\exists\,\mathsf{R}\,.\,\mathsf{C}$ for an arbitrary $\mathcal{ALCNH}_{R^+}$ concept term is found in an ABox (see [6] for the $\mathcal{ALCNH}_{R^+}$ tableaux calculus).[2] If lazy unfolding is applied, domain restrictions have to be considered w.r.t. the 'directly uses non-modal' relation in a special way.

Although it is possible to absorb a domain restriction such as the one expressed by $\exists\,\mathsf{anatomical\_part\_of\_ana\_heart}\,.\,\top \sqsubseteq \mathsf{ana\_heart\_p}$ into an equivalent inclusion $\neg\mathsf{ana\_heart\_p} \sqsubseteq \forall\,\mathsf{anatomical\_part\_of\_ana\_heart}\,.\,\bot$, lazy unfolding cannot be easily applied if an inclusion axiom for $\mathsf{ana\_heart\_p} \sqsubseteq \ldots$ exists. However, this is the case for UMLS. Hence, in order to apply the topological sorting optimization, incorporating domain restrictions into the tableaux calculus was necessary because meta constraints must not exist for topological sorting to be a valid optimization.

Note that, in principle, RACE also supports absorption of GCIs into inclusions $\neg\mathsf{A} \sqsubseteq \mathsf{C}_1$ (but only if no inclusion $\mathsf{A} \sqsubseteq \mathsf{C}_2$ or definition $\mathsf{A} \doteq \mathsf{C}_2$ exists). Some knowledge bases can only be handled effectively with $\neg\mathsf{A} \sqsubseteq \mathsf{C}_1$ absorptions.

In contrast to domain restrictions, range restrictions for roles do not introduce disjunctions. However, in a practical implementation it is advantageous to keep the number of internal data structures to be managed as small as possible. Therefore, range restrictions $\forall\,\mathsf{R}\,.\,\mathsf{C}$ are "considered" only if an existential restriction for R or a subrole of R are imposed for a certain individual i. These cases can also be easily detected.

2.3 Clustering

A problem with large knowledge bases is that the set of children of some concept names can get very large. Thus, the top search procedures exhibits worst case performance, i.e. the optimization techniques presented in [1] are not effective. Therefore, in the implementation of RACE a special clustering technique is employed.

If more than θ concept names are found to be children of a certain concept name, the θ children are grouped into a so-called bucket $\mathsf{A}_{\mathsf{new}}$, i.e. a (virtual)

[2]Implementation note: Due to our experiences, domain restrictions cannot be easily considered in the (recursive) encoding process for concepts. Encoding a concept term `(some r d)` as $\mathsf{C} \sqcap \exists\,\mathsf{R}\,.\,\mathsf{D}$ (with C being the domain restriction for R) would cause all kinds of trouble concerning the negation `(not (some r d))` of this term. The negation of this term would still be $\forall\,\mathsf{R}\,.\,\neg\mathsf{D}$ and not $\neg\mathsf{C} \sqcup \forall\,\mathsf{R}\,.\,\neg\mathsf{D}$ as suggested when `(some r d)` were encoded as $\mathsf{C} \sqcap \exists\,\mathsf{R}\,.\,\mathsf{D}$. So, a special treatment is necessary for these terms. But what if $\mathsf{C} \sqcap \exists\,\mathsf{R}\,.\,\mathsf{D}$ happens to be a concept term used in the knowledge base itself? Then, the negation definitely would be $\neg\mathsf{C} \sqcup \forall\,\mathsf{R}\,.\,\neg\mathsf{D}$. In this case, the encoding procedure can hardly guarantee uniqueness of the encoding result (which is essential for clash detection).

concept definition $\mathsf{A_{new}} \doteq \mathsf{A_1} \sqcup \ldots \sqcup \mathsf{A_\theta}$ is assumed and $\mathsf{A_{new}}$ is inserted into the subsumption lattice with $\mathsf{A_1} \ldots \mathsf{A_\theta}$ being its children. Note that bucket concepts $\mathsf{A_{new}}$ are virtual concepts in the sense that they are not mentioned in the set of children or parents of the concept names mentioned in a TBox.

Let us assume, a certain concept name A is inserted. Instead of testing whether each $\mathsf{A_i}$ ($\mathsf{i} \in \{1..\theta\}$) subsumes A during the top search phase, our findings suggest that it is more effective to initially test whether $\mathsf{A_{new}}$ does not subsume A using the model merging technique. Since in most cases, no subsumption relation can be found between any $\mathsf{A_i}$ and A, one test possibly replaces θ tests. On the other hand, if a subsumption relation indeed exists, then clustering introduces some overhead. However, since for almost all concept names only primitive concept definitions are included in the TBox, the model of $\neg\mathsf{A_{new}}$ being used for model merging is very simple because the model basically consists only of a set of negated concept names (see [3] for further details about model merging in RACE).

For best performance, the number of concepts to be kept in a bucket should depend on the number of subconcepts of the concept. However, this can hardly be estimated. Therefore, the following strategy is used. If more and more concept names are "inserted" into the subsumption lattice, the number of buckets increases as well. If a new bucket is to be created for a certain concept A and there are already σ buckets clustering the subconcepts of A, then two buckets (those buckets with the smallest number of children) are merged. Merging of buckets $\mathsf{A_{new}} \doteq \mathsf{A_1} \sqcup \ldots \sqcup \mathsf{A_n}$ and $\mathsf{B_{new}} \doteq \mathsf{B_1} \sqcup \ldots \sqcup \mathsf{B_m}$ means that the bucket $\mathsf{A_{new}}$ is "redefined" as $\mathsf{A_{new}} \doteq \mathsf{A_1} \sqcup \ldots \sqcup \mathsf{A_n} \sqcup \mathsf{B_1} \sqcup \ldots \sqcup \mathsf{B_m}$ and the bucket $\mathsf{B_{new}}$ is reused for the new bucket to be created (see above).[3] Whether hierarchical clustering techniques lead to performance improvements is subject to further research.

The current evaluation of clustering with buckets uses a setting with $\theta = 10$ and $\sigma = 15$.

2.4 Exploiting Disjointness Declarations

As has been discussed in [1], it is important to derive told subsumers for each concept name for marking and propagation processes. Besides told subsumers, RACE exploits also the set of "told disjoint concepts". In the 'heart' example presented above, ana_heart is computed as a told disjoint concept of ana_heart_p by examining inclusion axioms. If it is known that a concept B is a subsumer of a concept A then A cannot be a subsumee of the told disjoints of B. This kind of information is recorded (and propagated) with appropriate non-subsumer marks (see [1] for details about marking and propagation operations) such that this

[3]Note that due to subsequent merging operations, n and m need not be equal to θ.

information is not rediscovered with a model merging or even a tableaux-based subsumption test. Exploiting disjointness information has not been investigated in [1].

Traversing the subsumption lattice is also needed for ABox realization. The idea is to exploit disjointness information to speed-up the realization process as follows. Whenever an instance checking test $\mathsf{i : A}$ returns 'yes', it is obvious that i cannot be an instance of a concept that is a member of the set of told disjoint concepts of A. Thus, in the subsumption lattice, the told disjoint concepts are marked accordingly and an instance checking test for these concepts, which possibly involves an "expensive" ABox consistency test, is not necessary. Since a large number of instance checking tests must be performed, the exploitation of disjointness information is particularly effective for ABox realization (see [3] for experimental results).

2.5 Caching Policies

RACE supports different caching policies (see also [5] for caching in RACE). A cache for finding information about (sorted) sets of concepts is used for checking whether a set of concepts is satisfiable or unsatisfiable (so-called equal cache implemented as a hash table). Furthermore, a pair of caches for satisfiable as well as unsatisfiable concept sets is provided. These caches support queries concerning already encountered supersets and subsets of a given set of concepts [8]. For the UMLS benchmarks the (additional) equal cache had to be disabled in order to reduce space requirements.[4]

3 Empirical Results for UMLS TBox Classifications

The performance of the RACE system is evaluated with different versions of the UMLS knowledge base. UMLS-1 is a preliminary version that contains many inconsistent concept names. UMLS-1 consists of approximately 100,000 concept names and for almost all of them there exists a primitive concept definition $\mathsf{A} \sqsubseteq \mathsf{C}$ with C not being $\top$. In addition, in UMLS-1 80,000 role names are declared. Role names are arranged in a hierarchy.

UMLS-2 is a new version in which the reasons for the inconsistencies have been removed. The version of UMLS-2 we used for our empirical tests uses approximately 160,000 concept names and 80,000 roles.

Originally, the UMLS knowledge base has been developed with Loom 4.0 [11]. If Loom is given a cyclic definition for a certain concept, then Loom does not

[4]If the equal cache is enabled, it is the first reference. Only if a cache lookup fails, the superset or the subset caches are consulted. All retrieval results from the superset of subset caches are also entered into the equal cache. Due to our findings, the setting with maximum performance uses both an equal cache and a subset as well as a superset cache.

classify this concept (and the concepts which use this concept). Due to Loom's treatment of cycles, in [13] the cycle-causing concepts are placed in a so-called `:implies` clause, i.e. these restrictions are only asserted to individuals in an ABox via the rule mechanism. For the same reason, the UMLS reconstruction uses `:implies` for domain and range restrictions for roles, i.e. domain and range restrictions are only asserted in the ABox.

With RACE, none of these pragmatic distinctions are necessary. However, in order to mimic the Loom behavior and to test more than one TBox with RACE, for each of the knowledge base versions, UMLS-1 and UMLS-2, three different subversions are generated (indicated with letters a, b and c). Version 'a' uses axioms of the style presented above, i.e. the `:implies` parts are omitted for TBox classification (and coherence checking). In version 'b' the `:implies` part of the Loom knowledge base is indeed considered for classification by RACE. Thus, additional axioms of the following form are generated.

$$\begin{aligned}\textbf{ana\_heart} \sqsubseteq\ & \exists\, \mathsf{has\_developmental\_fo} \,.\, \mathsf{ana\_fetal\_heart} \sqcap \\ & \exists\, \mathsf{surrounded\_by} \,.\, \mathsf{ana\_pericardium}\end{aligned}$$

Version 'c' is the hardest version. Additional axioms provide domain and range restrictions for roles. For example, the following axioms are generated for $\mathsf{anatomical\_part\_of\_ana\_heart}$.

$$\begin{aligned}\exists\, \mathsf{anatomical\_part\_of\_ana\_heart} \,.\, \top &\sqsubseteq \mathsf{ana\_heart\_p} \\ \top &\sqsubseteq \forall\, \mathsf{anatomical\_part\_of\_ana\_heart} \,.\, \mathsf{ana\_heart}\end{aligned}$$

Thus, for the performance evaluation we have tested 6 different knowledge bases. All measurements have been performed on a Sun UltraSPARC 2 with 1.4 GByte main memory and Solaris 2.6. RACE is implemented in ANSI Common Lisp and for the tests Franz Allegro Common Lisp 5.0.1 has been used. The results are as follows:

Without clustering and topological sorting, classifying UMLS-1a can be done in approximately 11 hours (1636 concepts are incoherent). With clustering and topological sorting enabled, only 5.5 hours are necessary to compute the same result for UMLS-1a. The second version UMLS-1b requires 3.6 hours (with optimization) and 6.1 hours (without optimization). The reason for enhanced performance with more constraints is that in this version already 47855 concepts are inconsistent. With domain and range restrictions we found that even 60246 concepts are inconsistent. The computation times with RACE are 3.4 hours (with optimization) and 8.7 hours (without optimization). Up to 500 MBytes of memory are required to compute the classification results. For UMLS-1, checking TBox coherence (see above) requires approximately 10 minutes.

The new second version UMLS-2 contains an additional part of the UMLS and, therefore, is harder to deal with. Furthermore, there are no inconsistent concepts, i.e. classification is much harder because there are much more nodes in the

Table 1: Evaluation of the classifications of the UMLS-2 knowledge bases (T = topological sorting, C = clustering, R = runtime [hours:minutes], NST = number of subsumption tests, NM = number of cached models, MaxNC = maximal number of children, NB = number of buckets).

UMLS	**T**	**C**	**R**	**NST** ($\times 10^6$)	**NM** ($\times 10^3$)	**MaxNC**	**NB**
2a	on	on	10:13	232	251	26,874	3,110
	on	off	25:06	2,341	237	”	NA
	off	on	22:40	1,256	362	”	2,740
	off	off	31:26	2,796	281	”	NA
2b	on	on	10:11	232	251	26,874	3,110
	on	off	24:33	2,341	237	”	NA
	off	on	~22:00	~1,200	~360	”	~2,700
	off	off	30:18	2,796	281	”	NA
2c	on	on	14:53	222	255	21,298	3,273
	on	off	40:54	3,723	240	”	NA
	off	on	>50:00	?	?	”	?
	off	off	>50:00	?	?	”	NA

subsumption lattice. In UMLS-1, due to the large number of inconsistent concepts, the subsumption lattice is rather small because many concept “disappear” as synonyms of the bottom concept. For UMLS-2, checking TBox coherence (see above) requires between 15 and 50 minutes (2a: 16 min, 2b: 19 min, 2c: 51 min).

More detailed performance evaluations of RACE applied to UMLS-2 and TBox classification are presented in Table 1. In order to provide a machine-independent evaluation, not only runtimes are given but also the number of subsumption tests as well as other indicators are presented. It should be noted that the tableaux algorithm is needed only for computing models [9, 3]. In other words, all subsumption tests are decided by deep model merging tests [3]. A comparison of setting 1 (both topological sorting and clustering enabled) and setting 2 (clustering disabled) reveals that clustering is a very effective optimization technique for the UMLS-2 TBoxes. The result for setting 3 (topological sorting disabled) and UMLS2a supports the fact that topological sorting is also very effective.[5] The preliminary runtime result for setting 3 and UMLS3c is due to removed buckets. A bucket is removed if a member of this bucket gets a new parent assigned. This is very likely if topological sorting is disabled. If, in setting 4, both clustering and topological sorting are disabled, runtimes increase only to a limited extent.[6] Moreover, according to the evaluation results, the second version UMLS-2b does not require more computational resources than UMLS-2a (see the discussion about `:implies` from above). Only the incorporation of

[5]The result for setting 3 and UMLS2b is estimated due to lack of evaluation time, the test for setting 3 and UMLS3b is in progress at the time of this writing.

[6]The evaluation of setting 4 and UMLS2c is in progress at the time of this writing.

domain and range restrictions cause runtimes to increase. For UMLS-2 up to 800 MBytes are required. For other benchmark TBoxes (e.g. Galen with approx. 3000 concepts) our results suggest that there is neither overhead imposed by the clustering and topological sorting optimization techniques nor is there a significant gain to be observed.

In summary, the results for the UMLS TBoxes clearly demonstrate that clustering is only effective in conjunction with topological sorting establishing a quasi-definition order. The work reported here indicates that sound and complete description logic systems can now effectively deal with some instances of very large knowledge bases.

4 Conclusion

In this paper new optimization techniques which are essential for an initiative towards sound and complete high performance knowledge base classification are presented. Thus, fast classification even of simple but very large terminologies now has become possible with description logic systems based on sound and complete algorithms.

Even with all of the discussed optimization techniques enabled, dealing with 160,000 concept names reveals even slightly less optimal algorithms used for specific subproblems. For instance, in the Common Lisp implementation used for the performance tests described in this paper the algorithm for removing the duplicates found in a list exhibited quadratic time complexity. Hence, the library function (which worked perfectly in another Common Lisp implementation) had to be replaced by a more specific version using a hash table for indicating an element previously encountered (rather than linear search). With quadratic time complexity for the function that used the remove-duplicates function, no evaluation results could be computed within a reasonable amount of time. Other examples in which quadratic behavior had to be replaced by $n \log n$ behavior have been encountered as well. Thus, in a large system such as RACE, experiments with very large knowledge bases also provide feedback concerning lurking performance bottlenecks not becoming apparent when dealing with smaller knowledge bases such as Galen.

A final comment concerning the significance of the UMLS knowledge bases used for the empirical evaluation is appropriate. Even though all knowledge bases are "simple" in the sense defined above, a large number of "simple" concept definitions can be called the standard case in practical applications. The simple concepts might be used as a basis for more demanding concept definitions exploiting the real expressive power of $\mathcal{ALCNH}_{R^+}$. If description logics are to be successful in large-scale practical applications, being able to deal with large knowledge bases such as those based on the UMLS is mandatory. We would like to thank Stefan Schulz for making the UMLS reconstruction available.

References

[1] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making $\mathcal{KRIS}$ get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.

[2] A.G. Cohn, F. Giunchiglia, and B. Selman, editors. *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), Breckenridge, Colorado, USA, 2000.* Morgan Kaufmann, April 2000.

[3] V. Haarslev and R. Möller. Optimizing TBox and ABox reasoning with pseudo models. In this volume.

[4] V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In Lambrix et al. [10], pages 115–119.

[5] V. Haarslev and R. Möller. Consistency testing: The RACE experience. In *Proceedings International Conference TABLEAUX'2000*. Springer-Verlag, 2000.

[6] V. Haarslev and R. Möller. Expressive ABox reasoning with number restrictions, role hierachies, and transitively closed roles. In Cohn et al. [2], pages 273–284.

[7] V. Haarslev, R. Möller, and A.-Y. Turhan. RACE user's guide and reference manual version 1.1. Technical Report FBI-HH-M-289/99, University of Hamburg, Computer Science Department, October 1999. Available at URL http://kogs-www.informatik.uni-hamburg.de/~haarslev/publications/report-FBI-289-99.ps.gz.

[8] J. Hoffmann and J. Köhler. A new method to query and index sets. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence IJCAI-99*, pages 462–467. Morgan-Kaufmann Publishers, July 31 – August 6, 1999.

[9] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997.

[10] P. Lambrix et al., editor. *Proceedings of the International Workshop on Description Logics (DL'99), July 30 - August 1, 1999, Linköping, Sweden*, June 1999.

[11] R.M. MacGregor. A description classifier for the predicate calculus. In *Proc. of the Twelfth National Conference on Artificial Intelligence, AAAI-94*, pages 213–220, 1994.

[12] T. McCray, A and S.J. Nelson. The representation of meaning in the UMLS. *Methods of Information in Medicine*, 34(1/2):193–201, 1995.

[13] S. Schulz and U. Hahn. Knowledge engineering by large-scale knowledge reuse – Experience from the medical domain. In Cohn et al. [2], pages 601–610.