Description of the RACER System and its Applications

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

Abstract

RACER implements a TBox and ABox reasoner for the logic $\mathcal{SHIQ}$. RACER was the first full-fledged ABox description logic system for a very expressive logic and is based on optimized sound and complete algorithms.

1 Introduction

The description logic (DL) $\mathcal{SHIQ}$ [26] extends the logic $\mathcal{ALCNH}_{R^+}$ [14] by additionally providing qualified number restrictions and inverse roles. $\mathcal{ALCNH}_{R^+}$ was the logic supported by RACE (Reasoner for ABoxes and Concept Expressions), the precursor of RACER (Renamed ABox and Concept Expression Reasoner). Using the $\mathcal{ALCNH}_{R^+}$ naming scheme, $\mathcal{SHIQ}$ could be called $\mathcal{ALCQHI}_{R^+}$ (pronunciation: ALC-choir).

$\mathcal{ALCQHI}_{R^+}$ is briefly introduced as follows. We assume a set of concept names C, a set of role names R, and a set of individual names O. The mutually disjoint subsets P and T of R denote non-transitive and transitive roles, respectively ($R = P \cup T$). $\mathcal{ALCQHI}_{R^+}$ is introduced in Figure 1 using a standard Tarski-style semantics. The term $\top$ ($\bot$) is used as an abbreviation for $\mathsf{C} \sqcup \neg\mathsf{C}$ ($\mathsf{C} \sqcap \neg\mathsf{C}$).

If $\mathsf{R}, \mathsf{S} \in R$ are role names, then $\mathsf{R} \sqsubseteq \mathsf{S}$ is called a *role inclusion axiom*. A *role hierarchy* $\mathcal{R}$ is a finite set of role inclusion axioms. Then, we define $\sqsubseteq^*$ as the reflexive transitive closure of $\sqsubseteq$ over such a role hierarchy $\mathcal{R}$. Given $\sqsubseteq^*$, the set of roles $\mathsf{R}^{\downarrow} = \{\mathsf{S} \in R \mid \mathsf{S} \sqsubseteq^* \mathsf{R}\}$ defines the *sub-roles* of a role R. We also define the set $S := \{\mathsf{R} \in P \mid \mathsf{R}^{\downarrow} \cap T = \emptyset\}$ of *simple* roles that are neither transitive nor have a transitive role as sub-role.

Number restrictions are only allowed for simple roles. This restriction is motivated by a known undecidability result in case of an unrestricted syntax [25]. In concepts, instead of role names R (or S), inverse roles R^{-1} (or S^{-1}) may be used.

Syntax	**Semantics**
Concepts	
A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\neg \mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}}$
$\mathsf{C} \sqcap \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$
$\mathsf{C} \sqcup \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$
$\exists \mathsf{R} . \mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \exists \mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}}, \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\forall \mathsf{R} . \mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \forall \mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a}, \mathsf{b}) \in \mathsf{R}^{\mathcal{I}} \Rightarrow \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\exists_{\geq n} \mathsf{S} . \mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \ \|\{\mathsf{y} \mid (\mathsf{x}, \mathsf{y}) \in \mathsf{S}^{\mathcal{I}},\ \mathsf{y} \in \mathsf{C}^{\mathcal{I}}\}\| \ \geq \mathsf{n}\}$
$\exists_{\leq n} \mathsf{S} . \mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \ \|\{\mathsf{y} \mid (\mathsf{x}, \mathsf{y}) \in \mathsf{S}^{\mathcal{I}},\ \mathsf{y} \in \mathsf{C}^{\mathcal{I}}\}\| \ \leq \mathsf{n}\}$
Roles	
R	$\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

A is a concept name and $\|\cdot\|$ denotes the cardinality of a set. Furthermore, we assume that $\mathsf{R} \in R$ and $\mathsf{S} \in S$.

Axioms	
Syntax	Satisfied if
$\mathsf{R} \in T$	$\mathsf{R}^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}})^{+}$
$\mathsf{R} \sqsubseteq \mathsf{S}$	$\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$
$\mathsf{C} \sqsubseteq \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$

Assertions	
Syntax	Satisfied if
$\mathsf{a} : \mathsf{C}$	$\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
$(\mathsf{a}, \mathsf{b}) : \mathsf{R}$	$(\mathsf{a}^{\mathcal{I}}, \mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$

Figure 1: Syntax and Semantics of $\mathcal{ALCQHI}_{R^+}$.

If C and D are concepts, then $\mathsf{C} \sqsubseteq \mathsf{D}$ is a terminological axiom (*generalized concept inclusion* or *GCI*). A finite set of terminological axioms $\mathcal{T}_{\mathcal{R}}$ is called a *terminology* or *TBox* w.r.t. to a given role hierarchy $\mathcal{R}$.[1] An *ABox* $\mathcal{A}$ is a finite set of assertional axioms as defined in Figure 1.

An interpretation $\mathcal{I}$ is a *model* of a concept C (or *satisfies* a concept C) iff $\mathsf{C}^{\mathcal{I}} \neq \emptyset$ and for all $\mathsf{R} \in R$ it holds that iff $(x, y) \in \mathsf{R}^{\mathcal{I}}$ then $(y, x) \in (\mathsf{R}^{-1})^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ iff it satisfies all axioms in $\mathcal{T}$. See Figure 1 for the satisfiability conditions. An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ iff it is a model of $\mathcal{T}$ and satisfies all assertions in $\mathcal{A}$. Different individuals are mapped to different domain objects (unique name assumption).

2 Inference Services

In the following we define several inference services offered by RACER.

A *concept* is called *consistent* (w.r.t. a TBox $\mathcal{T}$) iff there exists a model of C (that is also a model of $\mathcal{T}$ and $\mathcal{R}$). An *ABox* $\mathcal{A}$ is *consistent* (w.r.t. a TBox

[1]The reference to $\mathcal{R}$ is omitted in the following if we use $\mathcal{T}$.

$\mathcal{T}$) iff $\mathcal{A}$ has model $\mathcal{I}$ (which is also a model of $\mathcal{T}$). A *knowledge base* ($\mathcal{T}$,$\mathcal{A}$) is called *consistent* iff there exists a model for $\mathcal{A}$ which is also a model for $\mathcal{T}$. A concept, ABox, or knowledge base that is not consistent is called *inconsistent*.

A concept D *subsumes* a concept C (w.r.t. a TBox $\mathcal{T}$) iff $\mathsf{C}^\mathcal{I} \subseteq \mathsf{D}^\mathcal{I}$ for all interpretations $\mathcal{I}$ (that are models of $\mathcal{T}$). If D subsumes C, then C is said to be *subsumed by* D.

Besides these basic problems, some additional inference services are provided by description logic systems. A basic reasoning service is to compute the subsumption relationship between concept names (i.e. elements from C). This inference is needed to build a hierarchy of concept names w.r.t. specificity. The problem of computing the most-specific concept names mentioned in $\mathcal{T}$ that subsume a certain concept is known as computing the *parents* of a concept. The *children* are the most-general concept names mentioned in $\mathcal{T}$ that are subsumed by a certain concept. We use the name *concept ancestors* (*concept descendants*) for the transitive closure of the parents (children) relation. The computation of the parents and children of every concept name is also called *classification* of the TBox. Another important inference service for practical knowledge representation is to check whether a certain concept name occcurring in a TBox is inconsistent. Usually, inconsistent concept names are the consequence of modeling errors. Checking the consistency of all concept names mentioned in a TBox without computing the parents and children is called a TBox *coherence check*.

An individual i is an *instance* of a concept C (w.r.t. a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$) iff $\mathsf{i}^\mathcal{I} \in \mathsf{C}^\mathcal{I}$ for all models $\mathcal{I}$ (of $\mathcal{T}$ and $\mathcal{A}$). For description logics that support full negation for concepts, the instance problem can be reduced to the problem of deciding if the ABox $\mathcal{A} \cup \{\mathsf{i}:\neg\mathsf{C}\}$ is inconsistent (w.r.t. $\mathcal{T}$). This test is also called *instance checking*. The most-specific concept names mentioned in a TBox $\mathcal{T}$ that an individual is an instance of are called the *direct types* of the individual w.r.t. a knowledge base $(\mathcal{T},\mathcal{A})$. The direct type inference problem can be reduced to subsequent instance problems (see e.g. [3] for details). The *retrieval* inference problem is to find all individuals mentioned in an ABox that are instances of a certain concept C. The set of *fillers* of a role R for an individual i w.r.t. a knowledge base $(\mathcal{T},\mathcal{A})$ is defined as $\{\mathsf{x} \mid (\mathcal{T},\mathcal{A}) \models (\mathsf{i},\mathsf{x}):\mathsf{R}\}$ where $(\mathcal{T},\mathcal{A}) \models ax$ means that all models of $\mathcal{T}$ and $\mathcal{A}$ also satisfy ax. The set of *roles* between two individuals i and j w.r.t. a knowledge base $(\mathcal{T},\mathcal{A})$ is defined as $\{\mathsf{R} \mid (\mathcal{T},\mathcal{A}) \models (\mathsf{i},\mathsf{j}):\mathsf{R}\}$.

As in other systems, there are some auxiliary queries supported: retrieval of the concept names or individuals mentioned in a knowledge base, retrieval of the set of roles, retrieval of the role parents and children (defined analogously to the concept parents and children, see above), retrieval of the set of individuals in the domain and in the range of a role, etc. As a distinguishing feature to other systems, which is important for many applications, we would like to emphasize that RACER supports multiple TBoxes and ABoxes. Assertions can be added to ABoxes after queries have been answered. In addition, RACER also provides support for retraction of assertions in particular ABoxes. The inference services supported by RACER for TBoxes and ABoxes are described in detail in [17].

3 The RACER Architecture

The ABox consistency algorithm implemented in the RACER system is based on the tableaux calculus of its precursor RACE [14]. For dealing with qualified number restrictions and inverse roles, the techniques introduced in the tableaux calculus for $\mathcal{SHIQ}$ [26] are employed.

However, optimized search techniques are required in order to guarantee good average-case performance. The RACER architecture incorporates the following standard optimization techniques: dependency-directed backtracking [32] and DPLL-style semantic branching (see [10] for an overview of the literature). Among a set of new optimization techniques, the integration of these techniques into DL reasoners for concept consistency has been described in [21]. The implementation of these techniques in the ABox reasoner RACER differs from the implementation of other DL systems, which provide only concept consistency (and TBox) reasoning. The latter systems have to consider only so-called "labels" (sets of concepts) whereas an ABox prover such as RACER has to explicitly deal with individuals (nominals). ABox optimizations are also explained in [13].

The techniques for TBox reasoning described in [3] (marking and propagation as well as lazy unfolding) are also supported by RACER. As indicated in [12], the architecture of RACER is inspired by recent results on optimization techniques for TBox reasoning [23], namely transformations of axioms (GCIs) [27], model caching [13] and model merging [21] (including so-called deep model merging and model merging for ABoxes [18]). RACER also provides additional support for very large TBoxes (see [15]). In addition, optimization techniques for dealing with qualified number restrictions [11, 20] are integrated into the RACER architecture. Another distinguishing feature of RACER is the adaptive use of optimization techniques by analyzing the input (knowledge base and queries). The advantage is that only one version of the RACER system is required for building applications. Although, e.g., for inverse roles the caching facility is automatically disabled, there is no performance penalty if the input does not contain inverse roles.

RACER is implemented in Common Lisp and is available for research purposes as a server program which can be installed under Linux and Windows (http://kogs-www.informatik.uni-hamburg.de/~race). Specific licenses are not required. Client programs can easily connect to a RACER DL server via a fast TCP/IP interface based on sockets. A client interface for Java is available.

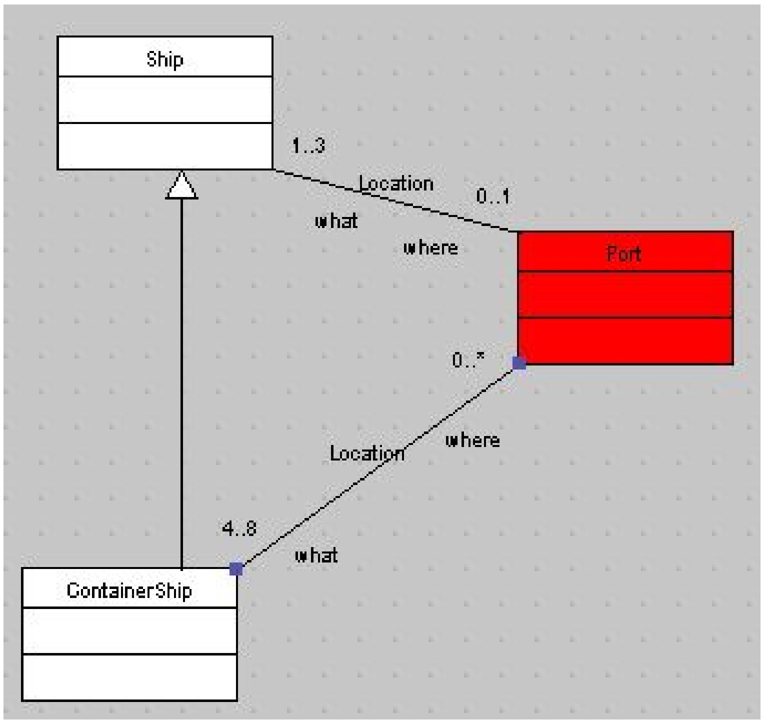

Figure 2: A UML diagram drawn with ArgoUML an verified with RACER.

4 Applications

The Java interface has been developed in order to support a TBox learning application (see [1]). An application of RACER for ontology engineering is described in [15]. RACER has also been used for UML verification [16] (see Figure 2). Ontologies can be built using powerful graphical tools (such as ArgoUML, http://argouml.tigris.org/) and saved as XML files in the XMI format. A compiler for translating model specifications in the XMI language into TBoxes has been developed for RACER. The semantics for UML specifications is ambiguous. We use an intuitive but "tentative" semantics. For the UML model presented in Figure 2, the following TBox is automatically generated.

$$\begin{aligned}
&\textbf{Ship} \sqsubseteq \exists_{\leq 1}\,\mathsf{what_Location_where}\,.\,\mathsf{Port}\\
&\textbf{ContainerShip} \sqsubseteq \mathsf{Ship}\\
&\textbf{Port} \sqsubseteq \exists_{\geq 1}\,\mathsf{what_Location_where}^{-1}\,.\,\mathsf{Ship} \sqcap\\
&\quad \exists_{\leq 3}\,\mathsf{what_Location_where}^{-1}\,.\,\mathsf{Ship} \sqcap\\
&\quad \exists_{\geq 4}\,\mathsf{what_Location_where}^{-1}\,.\,\mathsf{ContainerShip} \sqcap\\
&\quad \exists_{\leq 8}\,\mathsf{what_Location_where}^{-1}\,.\,\mathsf{ContainerShip}
\end{aligned}$$

A coherence check for the TBox obtained from the UML diagram in Figure 2 reveals the inconsistency of the concept port.

The theory behind another application of RACER in the domain of telecommunication systems is explained in [2]. An application of RACER for solving modal logic satisfiability problems is described in [13].

5 Concrete Domains

In addition to the language constructs mentioned above, reasoning about objects from other domains (so-called concrete domains, e.g. for the reals) is very important for practical applications as well. In [4] the description logic $\mathcal{ALC}(\mathcal{D})$ is investigated and it is shown that, provided a decision procedure for the concrete domain $\mathcal{D}$ exists, the logic $\mathcal{ALC}(\mathcal{D})$ is decidable.

Unfortunately, adding concrete domains (as proposed in the original approach) to expressive description logics might lead to undecidable inference problems. For instance, in [5] it is proven that the logic $\mathcal{ALC}(\mathcal{D})$ plus an operator for the transitive closure of roles can be undecidable if expressive concrete domains are considered. In [30] it is shown that $\mathcal{ALC}(\mathcal{D})$ with generalized inclusion axioms (GCIs) can be undecidable.

$\mathcal{ALCQHI}_{R^+}$ offers transitive roles but no operator for the transitive closure of roles. Even if GCIs were not allowed in $\mathcal{ALCQHI}_{R^+}$, $\mathcal{ALCQHI}_{R^+}$ with concrete domains would be undecidable (in general) because $\mathcal{ALCQHI}_{R^+}$ offers role hierarchies. Role hierarchies and transitive roles provide for the same expressivity as GCIs. With role hierarchies and transitive roles it is possible to (implicitly) declare a universal role, which can be used in combination with a value restriction to achieve the same effect as with GCIs. Decidability results can only be obtained for "trivial" concrete domains, which are hardly useful in practical applications. Thus, if termination and soundness of, for instance, a concept consistency algorithm are to be retained, there is no way extending an $\mathcal{ALCQHI}_{R^+}$ DL system such as RACE with concrete domains as in $\mathcal{ALC}(\mathcal{D})$ without losing completeness.

Thus, the logic supported by RACER can only be extended with concrete domain operators with limited expressivity. In order to support practical modeling requirements at least to some extent, we pursue a pragmatic approach by supporting only features (and no feature chains as in $\mathcal{ALC}(\mathcal{D})$, for details see [4]). The integration of concrete domains (e.g. linear inequalities between real numbers and constraints on the role fillers of a single individual) has been formally investigated for the expressive description logic $\mathcal{ALCNH}_{R^+}$ in [19]. We conjecture that the same techniques can also be applied for $\mathcal{ALCQHI}_{R^+}$. The RACER system supports concrete domains with release 1.6.

Although, in principle, concrete domains can be easily integrated since feature chains are not allowed, initial tests indicate that for real applications, *incremental* constraint satisfaction algorithms have to be explored for dealing with large search spaces. A Common Lisp implementation of the incremental constraint satisfaction techniques investigated in [28] has been developed. Optimization techniques described in [33] (model-merging with concrete domains) are currently integrated into the RACER system.

6 Summary and Outlook

In this paper we have described the RACER systems and its applications. The citations provide a survey of the state of the art in DL system implementation.

The UML verification with RACER is reminiscent of the utilization of the FaCT system [22] for the ICOM entity relationship modeling tool [9]. The application of description logics to formalize conceptual data modeling approaches (UML, XML, ER, etc.) has been investigated in [7] (see also subsequent publications of the authors). With the availability of the RACER system, practical experiments even with ABoxes are possible.[2] Future work might also consider reasoning about the dynamic behavior of UML objects using description logics.

As has been mentioned before, RACER will soon be extended with reasoning support for concrete domains (linear inequations over the reals). Other improvements that will be supplied with new versions of RACER are more clever blocking strategies in the presence of inverse roles (see [24]). Joint work with Ian Horrocks indicates that even for inverse roles caching strategies might be developed such that performance enhancements for knowledge bases with inverse roles can be achieved in the future.

References

[1] J. Alvarez. Tbox acquisition and information theory. In Baader and Sattler [6], pages 11–20.

[2] C. Areces, W. Bouma, and M. de Rijke. Description logics and feature interaction. In Lambrix et al. [29], pages 33–36.

[3] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems. *Applied Intelligence*, 2(4):109–138, 1994.

[2]Since, currently, RACER supports the unique name assumption for ABox reasoning, in some modeling approaches (e.g., for query containment) an extra level of non-determinism (e.g. the generation of multiple ABoxes) might be required.

[4] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Twelfth International Conference on Artificial Intelligence, Darling Harbour, Sydney, Australia, Aug. 24-30, 1991*, pages 452–457, August 1991.

[5] F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In Ohlbach and H.J., editors, *Proceedings, GWAI-92: Advances in Artificial Intelligence, 16th German Conference on Artificial Intelligence*, pages 132–143. Springer-Verlag, September 1992.

[6] F. Baader and U. Sattler, editors. *Proceedings of the International Workshop on Description Logics (DL'2000), Aachen, Germany*, August 2000.

[7] D. Calvanese, M. Lenzerini, and D. Nardi. *Logics for Databases and Information Systems*, chapter Description Logics for Conceptual Data Modeling, pages 229–263. Kluwer Academic Publishers, 1998.

[8] A.G. Cohn, F. Giunchiglia, and B. Selman, editors. *International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, April 2000.

[9] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modelling. In *7th Intl. Workshop on Knowledge Representation meets Databases (KRDB'00), Berlin, Germany*, 2000.

[10] J.W. Freeman. *Improvements to propositional satisfiability search algorithms.* PhD thesis, University of Pennsylvania, Computer and Information Science, 1995.

[11] V. Haarslev and R. Möller. Optimizing reasoning in description logics with qualified number restrictions. In this volume.

[12] V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In Lambrix et al. [29], pages 115–119.

[13] V. Haarslev and R. Möller. Consistency testing: The RACE experience. In R. Dyckhoff, editor, *Proceedings, Automated Reasoning with Analytic Tableaux and Related Methods*, number 1847 in Lecture Notes in Artificial Intelligence, pages 57–61. Springer-Verlag, April 2000.

[14] V. Haarslev and R. Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In Cohn et al. [8], pages 273–284.

[15] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel H. Levesque, editor, *International Joint Conference on Artificial Intelligence (IJCAI'2001), August 4th - 10th, 2001, Seattle, Washington, USA*. Morgan-Kaufmann, August 2001.

[16] V. Haarslev and R. Möller. RACER – ein Beschreibungslogiksystem für Wissensmanagement-Anwendungen. In *Professionelles Wissensmanagement: Erfahrungen und Visionen*. Shaker-Verlag, March 2001.

[17] V. Haarslev and R. Möller. RACER user's guide and reference manual version 1.5. Technical report, University of Hamburg, Computer Science Department, 2001.

[18] V. Haarslev, R. Möller, and A.-Y. Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In Massacci [31].

[19] V. Haarslev, R. Möller, and M. Wessel. The description logic $\mathcal{ALCNH}_{R+}$ extended with concrete domains. In Massacci [31].

[20] V. Haarslev, M. Timmann, and R. Möller. Combining tableaux and algebraic methods for reasoning with qualified number restrictions. In this volume.

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

[22] I. Horrocks. Benchmark analysis with fact. In *Proc. of the 4th Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX 2000)*, number 1847 in Lecture Notes In Artificial Intelligence, pages 62–66. Springer-Verlag, 2000.

[23] I. Horrocks and P. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, June 1999.

[24] I. Horrocks, U. Sattler, and S. Tobies. A PSpace-algorithm for deciding $\mathcal{ALCNI}_{R^+}$-satisfiability. LTCS-Report 98-08, LuFg Theoretical Computer Science, RWTH Aachen, Germany, 1998.

[25] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, September 1999.

[26] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic SHIQ. In David MacAllester, editor, *Proceedings of the 17th International Conference on Automated Deduction (CADE-17)*, number 1831 in Lecture Notes in Computer Science, Germany, 2000. Springer-Verlag.

[27] I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In Cohn et al. [8], pages 285–296.

[28] J. Jaffar, S. Michaylov, P.J. Stuckey, and R.H.C. Yap. The CLP($\Re$) language and system. *ACM Transactions on Programming Languages and Systems*, 14(3):339–395, July 1992.

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

[30] C. Lutz. The complexity of reasoning with concrete domains (revised version). LTCS-Report 99-01, LuFG Theoretical Computer Science, RWTH Aachen, 1999.

[31] F. Massacci, editor. *International Joint Conference on Automated Reasoning (IJCAR'2001), June 18-23, 2001, Siena, Italy.*, Lecture Notes in Artificial Intelligence. Springer-Verlag, June 2001.

[32] R.M. Stallman and G.J. Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9(2):135–196, 1977.

[33] A.-Y. Turhan and V. Haarslev. Adapting optimization techniques to description logics with concrete domains. In Baader and Sattler [6], pages 247–256.