RACER System Description

Volker Haarslev and Ralf Möller

University of Hamburg, Computer Science Department
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
http://kogs-www.informatik.uni-hamburg.de/~haarslev|moeller/

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. RACER also implements a decision procedure for modal logic satisfiability problems (possibly with global axioms).

1 Introduction

The description logic (DL) $\mathcal{SHIQ}$ [18] extends the logic $\mathcal{ALCNH}_{R^+}$ [9] 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.

The concept language of $\mathcal{ALCQHI}_{R^+}$ syntactically restricts the combination of number restrictions and transitive roles. Number restrictions are only allowed for simple roles. This restriction is motivated by a known undecidability result in case of an unrestricted syntax [17]. In concepts, instead of a role name R (or S), the inverse role R^{-1} (or S^{-1}) may be used.

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.

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

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

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

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 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 is a model of an ABox $\mathcal{A}$ w.r.t. a TBox 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 $\mathcal{T}$) 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 cer-

tain 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 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*.

If the description logic supports full negation, consistency and subsumption can be mutually reduced to each other since D subsumes C (w.r.t. a TBox $\mathcal{T}$) iff $\mathsf{C} \sqcap \neg\mathsf{D}$ is inconsistent (w.r.t. $\mathcal{T}$) and C is inconsistent (w.r.t. $\mathcal{T}$) iff C is subsumed by $\bot$ (w.r.t. $\mathcal{T}$). Consistency of concepts can be reduced to ABox consistency as follows: A concept C is consistent (w.r.t. a TBox $\mathcal{T}$) iff the ABox $\{\mathsf{a}:\mathsf{C}\}$ is consistent (w.r.t. $\mathcal{T}$). 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 $\mathcal{A}$). Again, for description logics that support full negation for concepts, the instance problem can be reduced to the problem of deciding if the ABox $\mathcal{A} \cup \{\mathsf{a}:\neg\mathsf{C}\}$ 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 types inference problems can be reduced to subsequent instance problems. The *retrieval* inference problem is to find all individuals mentioned in an ABox that are an instance 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 $\{\mathsf{x} \,|\, (\mathcal{T}, \mathcal{A}) \models (\mathsf{i}, \mathsf{x}):\mathsf{R}\}$ where $(\mathcal{T}, \mathcal{A}) \models ax$ means that all models of $\mathcal{T}$ and $\mathcal{A}$ are also models of ax. The set of *roles* between two individuals i and j w.r.t. a knowledge base $(\mathcal{T}, \mathcal{A})$ is defined as $\{\mathsf{R} \,|\, (\mathcal{T}, \mathcal{A}) \models (\mathsf{i}, \mathsf{j}):\mathsf{R}\}$.

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 [11].

3 The RACER Architecture

The ABox consistency algorithm implemented in the RACER system is based on the tableaux calculus of its precursor RACE [9]. For dealing with qualified number restrictions and inverse roles, the techniques introduced in the tableaux calculus for $\mathcal{SHIQ}$ [18] 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 [22] and

DPLL-style semantic branching (see [6] 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 [15]. 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 [8].

The techniques for TBox reasoning described in [3] (marking and propagation as well as lazy unfolding) are also supported by RACER. As indicated in [7], the architecture of RACER is inspired by recent results on optimization techniques for TBox reasoning [16], namely transformations of axioms (GCIs) [19], model caching [8] and model merging [15] (including so-called deep model merging and model merging for ABoxes [13]). RACER also provides additional support for very large TBoxes (see [10]).

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 connect to the RACER DL server via a very fast TCP/IP interface based on sockets. Client-side interfaces for Java and Common Lisp are available. A C/C++ interface is available soon.

4 Applications

An application of RACER for ontology engineering is described in [10]. The theory behind another application of RACER in the domain of telecommunication systems is explained in [2]. RACER has also be used for solving modal logic satisfiability problems [8] and for database integration tasks. The Java interface has been developed in order to support a TBox learning application (see [1]).

5 Outlook

The integration of techniques for representing "concrete domains" (e.g. linear inequalities between real numbers) on the role fillers of an individual has been investigated in [14]. In addition, optimization techniques for dealing with qualified number restrictions [12] will be integrated into RACER in the next release.

References

1. J. Alvarez. Tbox acquisition and information theory. In Baader and Sattler [4], pages 11–20.
2. C. Areces, W. Bouma, and M. de Rijke. Description logics and feature interaction. In Lambrix et al. [20], 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.
4. F. Baader and U. Sattler, editors. *Proceedings of the International Workshop on Description Logics (DL'2000), Aachen, Germany*, August 2000.

5. A.G. Cohn, F. Giunchiglia, and B. Selman, editors. *International Conference on Principles of Knowledge Representation and Reasoning (KR'2000)*, April 2000.
6. J.W. Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, University of Pennsylvania, Computer and Information Science, 1995.
7. V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In Lambrix et al. [20], pages 115–119.
8. 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.
9. V. Haarslev and R. Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In Cohn et al. [5], pages 273–284.
10. 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.
11. V. Haarslev and R. Möller. RACER user's guide and reference manual version 1.5. Technical report, University of Hamburg, Computer Science Department, 2001.
12. V. Haarslev and R. Möller. Signature calculus: Optimizing reasoning with number restrictions. Technical report, University of Hamburg, Computer Science Department, June 2001.
13. V. Haarslev, R. Möller, and A.-Y. Turhan. Exploiting pseudo models for tbox and abox reasoning in expressive description logics. In Massacci [21]. In this volume.
14. V. Haarslev, R. Möller, and M. Wessel. The description logic $\mathcal{ALCNH}_{R+}$ extended with concrete domains. In Massacci [21]. In this volume.
15. I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
16. I. Horrocks and P. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, June 1999.
17. 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.
18. 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.
19. I. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In Cohn et al. [5], pages 285–296.
20. P. Lambrix et al., editor. *Proceedings of the International Workshop on Description Logics (DL'99), July 30 - August 1, 1999, Linköping, Sweden*, June 1999.
21. 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.
22. 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.