# A Generator for Description Logic Formulas

Jan Hladik
Theoretical Computer Science, TU Dresden
hladik@tcs.inf.tu-dresden.de

**Abstract**

We introduce a schema for generating random formulas for different description logics, which extends an existing pattern for modal logics. Using the DL reasoners FaCT and Racer, we test the difficulty of these formulas, and it turns out that the properties that make a formula in an expressive DL hard are quite different from those known for $\mathcal{ALC}$ formulas.

## 1 Introduction

Empirical evaluation of description logic (DL) reasoners with benchmark formulas serves many purposes. Firstly, it enables the programmer to verify that changes he makes to his program improve efficiency. Secondly, it makes comparisons between different system possible. Thirdly, the question is still open why existing DL reasoners perform much better "in practice" than their worst-case complexity suggests, and the observation which kinds of formulas are easy or hard for a reasoner can help answer it. Finally, empirical tests allow for the verification of the correctness of the reasoner's results, if the satisfiability/unsatisfiability of the corresponding formula is known.

Comparison of the efficiency of different DL systems has taken place frequently during the recent years [BH98, Mas99, MD00, PSS03]. In the absence of existing benchmark formulas for DLs, these comparisons used benchmark formulas for modal logics (ML), which allowed for the comparison DL reasoners with SAT-based systems like *SAT [GGT02] or resolution-based systems like MSpass [HS00] for modal logics, as in the TANCS-2000 comparison [MD00].

However, using ML benchmark formulas to test DL reasoners also has a significant shortcoming: it only allows for the test of those language features that are common to DL and ML. In practice, this restricted the logic to **K** in TANCS-98 and -99 [BH98, Mas99] and $\mathbf{K}^-$ (**K** with inverse modality) in TANCS-2000 [MD00]. Thus, even if most DL systems nowadays can handle logics which are significantly more expressive than the modal logics mentioned, these additional features are not evaluated in the comparison tests. For example, *qualifying number restrictions (QNR)* [HST99] are a very important feature of expressive DLs, whereas the corresponding construct in ML *(graded modalities)* receives less attention and does therefore not appear in the formulas used in the above-mentioned comparisons. Thus, they cannot tell us if one reasoner handles QNR significantly better than another one, and they do not allow

the programmer to thoroughly test his implementation of the corresponding function for correctness.

In this paper, we present a schema for randomly generating DL formulas, which is based on the pattern described in [MD00, PSS03], but extends this to cover a much larger range of features available in current description logics, like QNR, transitive roles, or nominals. Since many language features can also be disabled by the user, our schema covers a wide range of DLs from $\mathcal{EL}$ and $\mathcal{FL}_0$ to $\mathcal{ALCQIO}_{R^+}$.

**Remark.** Because of the close relationship between modal and description logics, we will occasionally use modal logic terminology to describe features of description logics, e.g. for the concept $\forall R.\exists S.\forall R.(C \sqcap (D \sqcup \neg E))$, we will call $\forall R.\exists S.\forall R$ the *modal* part and $(C \sqcap (D \sqcup \neg E))$ the *propositional* part. We will also use the terms "formula" and "concept" synonymously.

## 2 Random DL formulas

The *KCNF* benchmark formulas, which were used in most of the comparisons mentioned above [Mas99, MD00, PSS03], are based on a schema developed by Giunchiglia and Sebastiani [GS96], which was subsequently analysed and improved by Hustadt and Schmidt [HS97], adapted by Massacci for the Tableaux system comparison (TANCS) [Mas99] and most recently extended by Patel-Schneider and Sebastiani [PSS03] for a comparison between *SAT and DLP [PS98]. The following definition is according to [PSS03].

**Definition 1 (KCNF)** Let $\{A_1, \ldots, A_N\}$ be a set of propositional variables and $\{\Box_1, \ldots, \Box_m\}$ be a set of modal operators. A *KCNF formula is* a conjunction of KCNF clauses. A *KCNF clause* is a disjunction of literals. A *literal* is an atom or its negation. There are two kinds of *atoms*: a propositional atom is a propositional variable, and a modal atom has the form $\Box_i C$, where $C$ is a KCNF clause.

A KCNF formula is characterised by the following parameters: the number $L$ of top-level conjuncts, the size $C$ of the clauses, the number $N$ of propositional variables, the number $m$ of modal variables, the maximum quantification depth $d$, and the probability $p$ of an atom occurring at depth $< d$ being purely propositional.

In order to adapt this pattern to produce formulas suited for state-of-the-art DL reasoners, we added several features: we allow for qualifying number restrictions, transitive as well as inverse roles [HST99], and nominals (see e.g. [Tob00]). Instead of clauses, we can create arbitrary Boolean combinations of literals. (At the top level, there is still a conjunct because otherwise a simple satisfiable disjunct could make a formula trivially satisfiable.) Moreover, the parameters $d$, $L$ and $C$ can be treated as a fixed or as a maximum value, i.e. $C = 5$ can mean either "every clause is of size 5" or "the size of every clause is between 1 and 5." Most syntactic elements can be switched off, so that also inexpressive DLs are supported.

We also modified some properties of the KCNF pattern: in [PSS03], most formulas for the parameters $d = 2$ and $p = 0$ (i.e. the depth 2 is reached in every conjunct)

cannot be solved within a timeout of 1000 seconds. As this strongly restricts the formulas which can reasonably be generated by this method, we imposed the additional constraint that every clause can contain at most one modal literal, which means that the size of the formula only increases linearly with $d$ instead of exponentially.

Under this condition, determining the depth of a random formula using the parameter $p$ would result in different probabilities for each depth. Since we consider an equal probability to be desirable, we removed the parameter $p$ and gave each depth between 1 and $d$ the same probability (the user has the option to enforce the maximum depth, which corresponds to setting $p = 0$).

During our tests, we observed that formulas that have propositional atoms only at the maximum quantification depth (i.e. formulas consisting of a purely modal and a purely propositional part) are much harder than formulas with propositional atoms at every depth (see Section 4). Therefore, we included the option of allowing propositional atoms at every depth, only at maximum depth, or randomly.

Hustadt and Schmidt [HS97] argued that the possibility to create trivially satisfiable or unsatisfiable clauses can lead to the creation of formulas which do not provide useful results, and thus the creation of a clause like $A \sqcup \neg A \sqcup B$ must be avoided. However, our propositional formulas are not necessarily clauses (disjunctions), but random combinations of conjunctions and disjunctions, and a formula $A \sqcup (\neg A \sqcap B)$ is not trivial. Therefore we do not enforce that all concept names in a propositional formula are distinct. This leads us to the following definition:

**Definition 2 (RDL)** A *random DL (RDL) formula* $D$ of depth $d$ is a conjunction $E_1 \sqcap E_2 \sqcap \ldots \sqcap E_L$. Each conjunct $E$ of depth $d > 0$ has the format $P \bigcirc \mathbf{Q}R.E'$, where $\bigcirc$ is a Boolean junctor, $\mathbf{Q}$ is a quantifier $(\forall, \exists, \geqslant n, \leqslant n)$, $R$ is a role, $E'$ is a concept of depth $d - 1$, and $P$ is a propositional formula: it consists of $C$ concept names, connected with randomly selected junctors. A conjunct of depth 0 only consists of the propositional formula $P$.

If inverse roles are permitted, each occurrence of a role name is inverse with a probability of 50%. If transitive roles are permitted, each role is transitive with a probability of 50%. If primitive negation is permitted, each concept name is negated with a probability of 50%. If propositional formulas are only permitted at maximum quantification level, the propositional formula $P$ only appears at depth 0.

Thus, an RDL formula is characterised by the following parameters: the set $S_Q$ of available quantifiers (among $\exists, \forall, \leqslant, \geqslant$) the set $S_B$ of available Boolean operators (among $\neg, \sqcap, \sqcup$) the maximum value $q$ for the number $n$ appearing in QNR, the number of nominals $O$, the probability $o$ of a propositional atom being a nominal, the possibility of transitive and inverse roles, and the parameters $L$, $C$, $N$, $m$ and $d$ of KCNF formulas.

For example, a formula for the parameter set $L = 2$, $d = 2$, $C = 3$, $S_Q = \{\forall, \exists, \geqslant\}$, $S_B = \{\neg, \sqcup, \sqcap\}$, $q = 5$, $N = 10$, $m = 2$, $O = 1$, propositional formulas at every level and the possibility of inverse roles, might look like this:

$$\begin{array}{ll} \exists R1 & (C5 \sqcap (\neg C9 \sqcup C4) \sqcap (\geqslant 4\ R1^- \ (C2 \sqcup \neg C8 \sqcup \neg C6))) \quad \sqcap \\ \forall R2^- & (\neg C1 \sqcap (C5 \sqcup C2) \sqcup \forall R2((C3 \sqcup C5) \sqcap \neg C5)) \end{array}$$

## 3 The Generator

We will now briefly describe the formula generator, which was implemented by Raj Mudunuri in the language TK as a student project. The user interface (Figure 1) essentially consists of masks for the parameters described in Section 2.

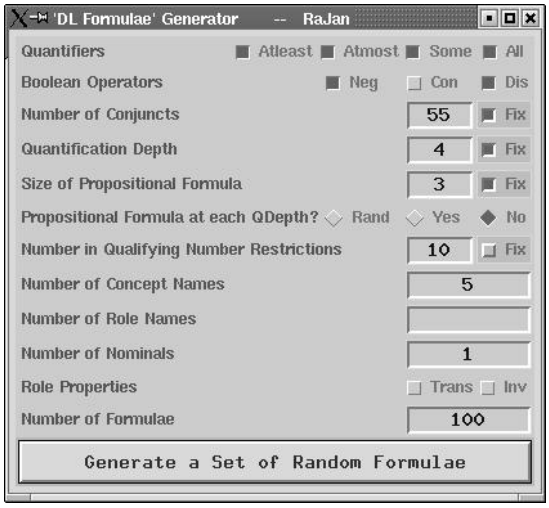



Figure 1: User interface of the generator.

If the "Fix" checkbox next to the parameters $L$, $d$, $C$ and $q$ is checked, the corresponding value is treated as the exact parameter, otherwise it is the maximum possible one. When the "Generate" button is clicked, the desired number of formulas is generated and written into a file in KRSS syntax [PSS93]. A second file is created with the definitions of primitive concepts and roles, and a third one containing the parameters.

## 4 Empirical Results

Firstly, we wanted to try if, using appropriate settings, it is possible to produce formulas resembling the KCNF ones, and if FaCT [Hor98] and Racer [HM01] would display a behaviour similar to KSAT and DLP in [PSS03]. In order to achieve this, we chose only $\forall$ and $\exists$ as quantifiers, and $\neg$ and $\sqcup$ as Boolean operators. We fixed the other parameters as follows: $m = 1$, $C = 3$, $N \in \{3, 6, 9\}$. For the modal depth $d$, we chose the value 4 in order to test if a limitation on the width makes a larger depth possible (see Section 2). Since we observed that having a propositional formula not at every level of quantification, but only at the maximum one, leads to much more challenging formulas (see below), we chose the latter option. In order to obtain a homogeneous set of formulas for each setting, the parameters are $L$, $d$ and $C$ are fixed.

Our results are presented in Figure 2 in a format similar to the one used in [PSS03]: the first diagram shows the percentage of satisfiable and unsatisfiable formulas in

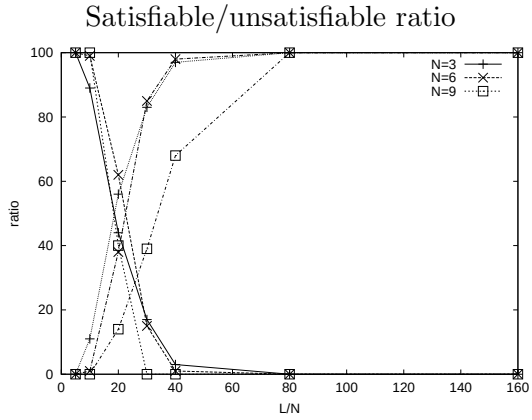



Figure 2: KCNF-like formulas

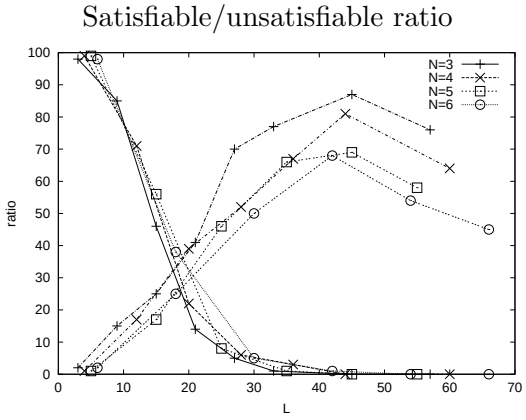
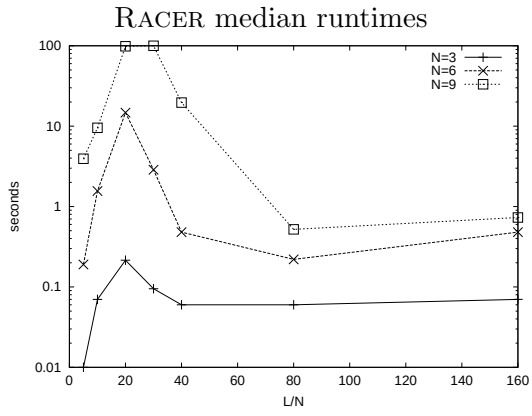
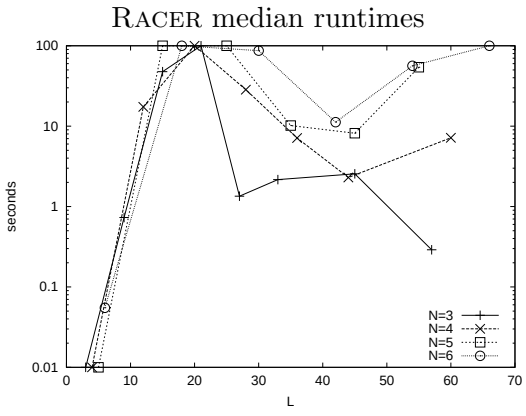
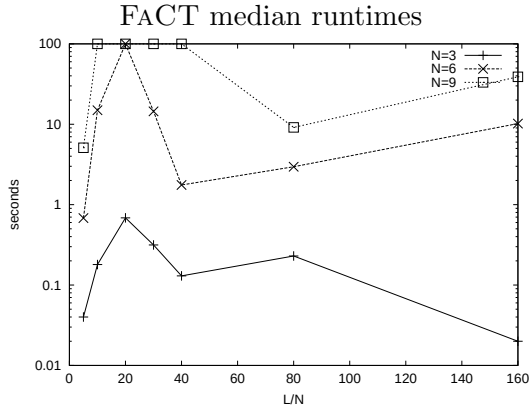
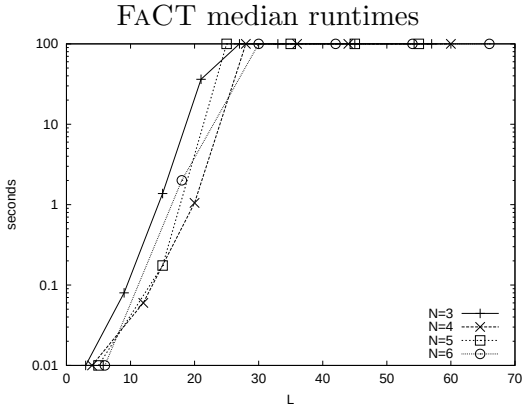



Figure 3: QNR formulas




relation to $L/N$,[1] and in the graphs below we show the runtimes of RACER and FACT for different values of $N$. For each measuring point, we tested 100 formulas, with a timeout of 100 seconds per formula. The first two experiments were performed on a Pentium 4 (2.4GHz) with 2GB RAM, the remaining two on a Pentium 4 (1.7GHz) with 512MB RAM. We used FACT version 2.31.7 and RACER version 1.6.7.

[1] If both numbers do not add to 100 for a single value, this indicates that not all formulas could be tested within the timeout.

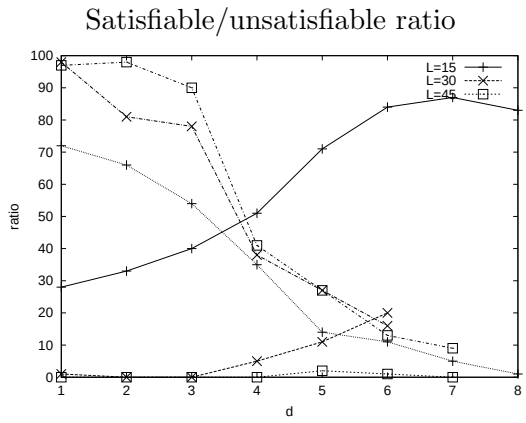



Figure 4: Formulas with different depth

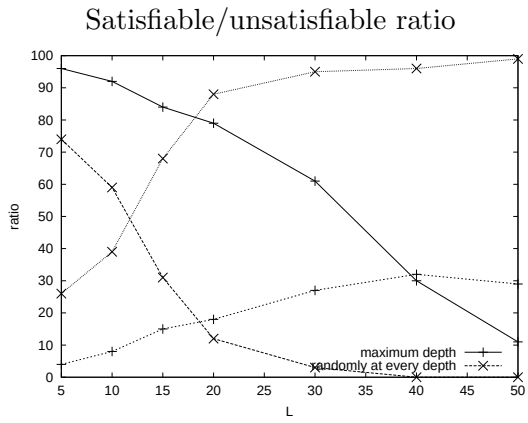
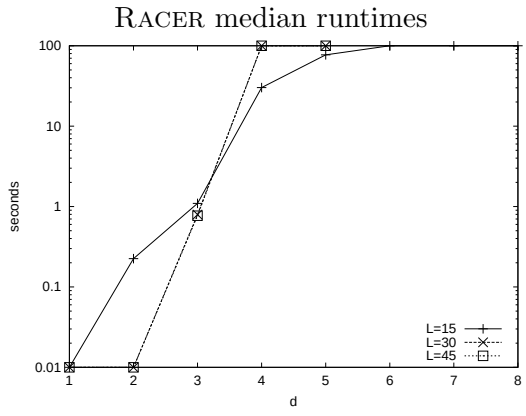
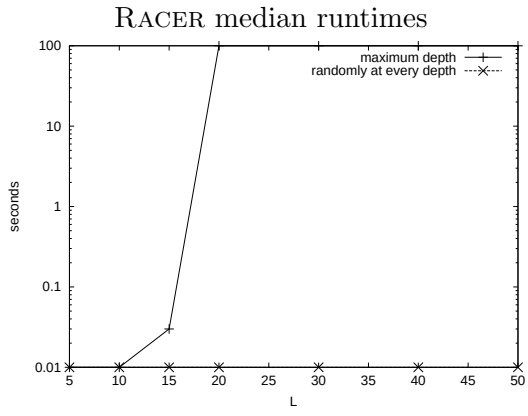
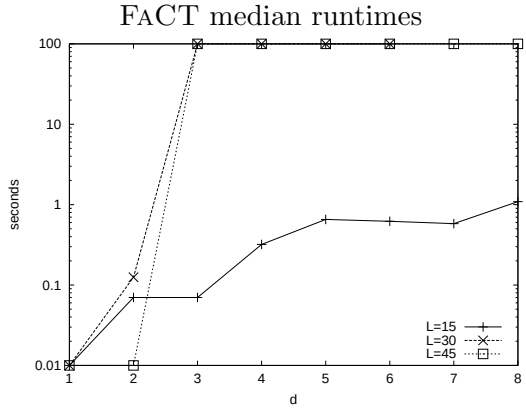
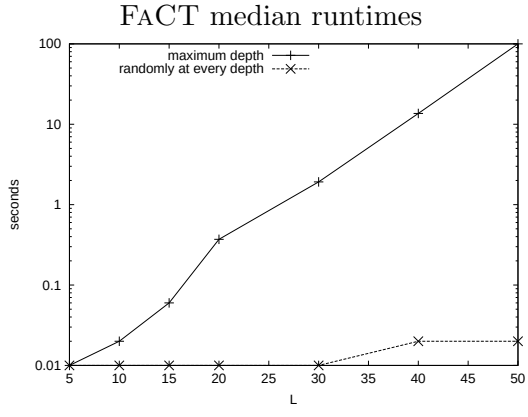



Figure 5: Expressive formulas




We can observe that the ratio of satisfiable and unsatisfiable formulas depends primarily on the ratio $L/N$: the larger the value of this fraction is, the more formulas are unsatisfiable. The crossover point is near the value 20, and this is also the region for which the calculation takes the longest time. A larger value of $N$ does not have an influence on these characteristics, but it makes a formula harder. This is very similar to the observations made in [PSS03], and therefore we believe it is justified to say that, using appropriate settings, we can reproduce the results made with the KCNF

formulas by Patel-Schneider and Sebastiani. However, we observe a modest increase in computation time for very large values of $L/N$. Comparing the two reasoners, one can see that RACER is approximately one order of magnitude faster than FACT.

In order to test the influence of the DL features added to the formulas, we allowed for qualifying number restrictions with a maximum value of $q = 10$ for our next experiment (Figure 3). It turns out that with these settings, the location of crossover point between satisfiable and unsatisfiable formulas does not depend on the ratio $L/N$, instead it is always near $L = 15$ (note the different labelling of the x-axis), i.e. much sooner than for the KCNF-like formulas. RACER again displays a peak in computation time in this area, but there is another significant increase for values of $L > 40$. The behaviour of FACT is completely different: for values of $L < 25$, it is between one and two orders of magnitude faster than RACER, but for values of $L > 30$, it can never solve more than half of the formulas, which results in a median time of 100 seconds. This behaviour indicates that FACT is faster for satisfiable formulas and RACER is faster for unsatisfiable ones. The parameter $N$ has little influence on both the satisfiability ratio and the computation time, which leads us to the conclusion that for formulas including QNR, these two values are determined by the modal part of the formula instead of the propositional part.

In our next test, which is intended to measure the influence of the quantification depth, we therefore kept the parameter $N$ fixed at the value 5 and varied $d$ between 1 and 8 for $L \in \{15, 30, 45\}$ (Figure 4). Note that for this test, the ratio of satisfiable formulas increases from the left to the right end of the diagram, unlike in the other experiments. For $L \in \{30, 45\}$, most formulas are either very easy or very hard to solve. For $L = 15$, the difficulty increases continuously with the depth, independently of the crossover point. In this test, FACT is significantly faster than RACER.

In our final experiment, we also added conjunction and inverse roles, so that we generated formulas for the logic $\mathcal{ALCQI}$. The parameters are $N = 5$, $d = 6$, and we vary $L$ between 5 and 50. We also generated a formula set with the same parameters, but propositional formulas appearing not only at the maximum depth, but randomly at every depth. The results are shown in Figure 5. For the benchmark set with propositional formulas appearing only at maximum depth, the graph for FACT is nearly linear, which indicates an exponential increase of the runtime in relation to $L$, whereas RACER takes much more time if there are more than 15 conjuncts. With propositional formulas randomly at every depth, the formulas become trivial for both reasoners.

## 5 Conclusion and Further Work

We have developed and implemented a generation schema for DL formulas. Using existing DL reasoners, we have tested several sets of formulas to determine the ratio of satisfiable vs. unsatisfiable formulas and the difficulty (measured in terms of calculation time), and our results show that the logics $\mathcal{ALC}$ and $\mathcal{ALCQ}$ differ significantly with respect to the parameters which determine these two properties. Firstly, while the satisfiability of an $\mathcal{ALC}$ formula depends on the ratio of the size of the formula (more precisely, the number of conjuncts) and the number of available concept names,

only the size has influence for an $\mathcal{ALCQ}$ formula. Secondly, while the hardest $\mathcal{ALC}$ formulas are found near the crossover point, the difficulty of an $\mathcal{ALCQ}$ formula increases continuously with its size. We also noticed different patterns in the behaviour of the reasoners used: FaCT is faster for satisfiable formulas and can deal better with formulas of a large depth, whereas Racer is better for unsatisfiable formulas and for those formulas which resemble the KCNF pattern for modal logic.

We are currently extending the formula generator to TBoxes. Our aim is to generate TBoxes with properties resembling to existing "real-life" knowledge bases like GALEN and to test which properties have to be modified in order to make a TBox harder to classify.

## References


[BH98] P. Balsinger and A. Heuerding. Comparison of theorem provers for modal logics. In H. de Swart, editor, *Proceedings of TABLEAUX-98*, number 1397 in LNAI. Springer-Verlag, 1998.

[Dyc00] R. Dyckhoff, editor. *Proceedings of TABLEAUX 2000*, volume 1847 of *LNAI*. Springer-Verlag, 2000.

[GGT02] E. Giunchiglia, F. Giunchiglia, and A. Tacchella. SAT based decision procedures for classical modal logics. *Journal of Automated Reasoning*, 28:143–171, 2002.

[GS96] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures — the case study of modal K. In M. A. McRobbie and J. K. Slaney, editors, *Proceedings of CADE-96*, volume 1104 of *LNAI*. Springer-Verlag, 1996.

[HM01] V. Haarslev and R. Möller. RACER system description. In *Proceedings of IJCAR-01*, volume 2083 of *LNAI*. Springer-Verlag, 2001.

[Hor98] I. Horrocks. Using an expressive description logic: FaCT or fiction? In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Proceedings of KR-98*. Morgan Kaufmann Publishers, 1998.

[HS97] U. Hustadt and R. A. Schmidt. On evaluating decision procedures for modal logic. Technical Report MPI-I-97-2-003, Max-Planck-Institut für Informatik, Saarbrücken, 1997.

[HS00] U. Hustadt and R. A. Schmidt. MSPASS: Modal reasoning by translation and first-order resolution. In Dyckhoff [Dyc00].

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proceedings of LPAR '99*, number 1705 in LNAI. Springer-Verlag, 1999.

[Mas99] F. Massacci. Design and results of the Tableaux-99 non-classical (modal) systems comparison. In N. Murray, editor, *Proceedings of TABLEAUX-99*, volume 1617 of *LNAI*. Springer-Verlag, 1999.

[MD00] F. Massacci and F. M. Donini. Design and results of TANCS-2000. In Dyckhoff [Dyc00].

[PS98] P. F. Patel-Schneider. DLP system description. In E. Franconi, G. De Giacomo, R. M. MacGregor, W. Nutt, and C. A. Welty, editors, *Proceedings of DL'98*, CEUR Proceedings, 1998.

[PSS93] P. F. Patel-Schneider and B. Swartout. Description logic knowledge representation system specification from the KRSS group of the ARPA knowledge sharing effort. Technical report, AI principles research department, AT&T Bell Laboratories, 1993. Available at `http://dl.kr.org`.

[PSS03] P.F. Patel-Schneider and R. Sebastiani. A new general method to generate random modal formulae for testing decision procedures. *Journal of Artificial Intelligence Research*, 18:351–389, 2003.

[Tob00] S. Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000.