

# HyLoRes: Direct Resolution for Hybrid Logics

## (System Report)

Carlos Areces    Juan Heguiabehere  
Computational and Applied Logic Group  
Faculty of Science. University of Amsterdam  
`{carlos, juanh}@science.uva.nl`

## 1 Introduction

Hybrid languages are modal languages that allow direct reference to the elements of the model. Already the basic hybrid language ( $\mathcal{H}(@)$ ) which extends the basic modal language with the addition of nominals ( $i, j, k, \dots$ ) and satisfiability operators ( $@_i, @_j, @_k, \dots$ ), obviously increases the expressive power of the language as we can now explicitly check whether the actual point of evaluation is some specific, named point in the model ( $w \Vdash i$ ), and whether a named point satisfies a given formula ( $w \Vdash @_i \varphi$ ). More interestingly, this extended expressive power permits also the definition of very elegant decision algorithm, where nominals and  $@$  play, inside the object language, the role of labels, or prefixes, which are usually needed during the construction of a proof in the modal setup (see, e.g. [8, 4]). And all these features we get with no increase in complexity (up to a polynomial): the complexity of the satisfiability problem for  $\mathcal{H}(@)$  is the same as for the basic modal language, PSPACE.

When we move into very expressive hybrid languages containing binders, we obtain, of course, an impressive boost in expressivity, but we also usually move beyond the boundaries of decidability. Classical binders like  $\forall$  and  $\exists$  (together with  $@$ ) make the logic as expressive as first-order logic (see [2]). On the other hand, more “modal” binders, like  $\downarrow$  which binds a variable only to the *actual point of evaluation*, provide extended expressivity, but remain below the expressive power of first-order logic. Actually, the language  $\mathcal{H}(@, \downarrow)$  is extremely well behaved: it has a very strong form of interpolation, it can be characterized in many different and interesting ways (in particular, it coincides in expressive power with the Bounded Fragment of first-order logic [3]), and has interesting decidable fragments (for example, the fragment where  $\downarrow$  never appears nested). See [3, 2].

We refer to [9] for a more detailed introduction to hybrid languages, and to the Hybrid Logic Site at <http://www.hylo.net> for a historical overview and a broad on-line bibliography. In recent years, an important number of new theoretical results concerning axiomatizability, interpolation, expressive power, complexity, etc. for hybrid logics have been obtained. The next natural step is to develop provers that can handle these languages.

HyLoRes is a direct resolution prover for hybrid logics (it can handle satisfiability of sentences in  $\mathcal{H}(@, \downarrow)$ ), implementing the algorithm presented in [5]. The implementation we present in this system description is very preliminary (this is version 1.0 of the prover), and it is not meant to compete with state of the art provers for modal and description logics like DLP [18], FaCT [13] or RACER [12]. Contrasting it with the provers mentioned before, the most interesting feature of HyLoRes is that it is not based on a tableau algorithm but on (direct) resolution. In particular, HyLoRes implements a version of the “given clause”

algorithm (see, e.g. [20]), which is nowadays the standard skeleton of the most powerful first-order provers. Notice also that in contrast to translation based provers like MSPASS [14], it performs resolution directly on the modal (or hybrid) input, with no translation into large background logics.

In this system description we first introduce the syntax and semantics of the language  $\mathcal{H}(@, \downarrow)$ , briefly list the rules of the direct resolution calculus presented in [5], and provide some examples. We then discuss details on how these rules were implemented in HyLoRes, and close with a discussion on the further developments we intend to integrate into the prover.

## 2 The hybrid language $\mathcal{H}(@, \downarrow)$

**Definition 2.1** (Syntax). Let  $\text{REL} = \{R_1, R_2, \dots\}$  be a countable set of *relational symbols*,  $\text{PROP} = \{p_1, p_2, \dots\}$  a countable set of *propositional variables*,  $\text{NOM} = \{i_1, i_2, \dots\}$  a countable set of *nominals*, and  $\text{SVAR} = \{x_1, x_2, \dots\}$  a countable set of *state variables*. We assume that these sets are pairwise disjoint. We call  $\text{SSYM} = \text{NOM} \cup \text{SVAR}$  the set of *state symbols*, and  $\text{ATOM} = \text{PROP} \cup \text{NOM} \cup \text{SVAR}$  the set of *atoms*. The well-formed formulas of the hybrid language  $\mathcal{H}(@, \downarrow)$  in the signature  $\langle \text{REL}, \text{PROP}, \text{NOM}, \text{SVAR} \rangle$  are

$$\text{FORMS} := \top \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [R]\varphi \mid @_s\varphi \mid \downarrow x.\varphi,$$

where  $a \in \text{ATOM}$ ,  $x \in \text{SVAR}$ ,  $s \in \text{SSYM}$ ,  $R \in \text{REL}$  and  $\varphi, \varphi_1, \varphi_2 \in \text{FORMS}$ .  $\mathcal{H}(@)$  is the sublanguage of  $\mathcal{H}(@, \downarrow)$  without state variables and  $\downarrow$ .

Note that the above syntax is simply that of ordinary (multi-modal) propositional logic extended with clauses for  $@_s\varphi$  and  $\downarrow x_j.\varphi$ . The difference between nominals and state variables is simply this: nominals cannot be bound by  $\downarrow$ , whereas state variables can. The notions of *free* and *bound* state variable are defined as in first-order logic, with  $\downarrow$  as the only binding operator. A *sentence* is a formula containing no free state variables. Now for the semantics, in the rest of the paper we assume fixed a signature  $\langle \text{REL}, \text{PROP}, \text{NOM}, \text{SVAR} \rangle$ .

**Definition 2.2** (Semantics). A (hybrid) *model*  $\mathcal{M}$  is a triple  $\mathcal{M} = \langle M, \{R_i\}, V \rangle$  such that  $M$  is a non-empty set,  $\{R_i\}$  is a set of binary relations on  $M$ , and  $V : \text{PROP} \cup \text{NOM} \rightarrow \text{Pow}(M)$  is such that for all nominals  $i \in \text{NOM}$ ,  $V(i)$  is a singleton subset of  $M$ . An *assignment*  $g$  for  $\mathcal{M}$  is a mapping  $g : \text{SVAR} \rightarrow M$ . Given an assignment  $g$ , we define  $g_m^x$  (an *x-variant* of  $g$ ) by  $g_m^x(x) = m$  and  $g_m^x(y) = g(y)$  for  $x \neq y$ .

Let  $\mathcal{M} = \langle M, \{R_i\}, V \rangle$  be a model,  $m \in M$ , and  $g$  an assignment. For any atom  $a$ , let  $[V, g](a) = \{g(a)\}$  if  $a$  is a state variable, and  $V(a)$  otherwise. Then the *satisfiability relation* is defined as follows

$$\begin{array}{ll} \mathcal{M}, g, m \Vdash \top & \text{always} \\ \mathcal{M}, g, m \Vdash a & \text{iff } m \in [V, g](a), a \in \text{ATOM} \\ \mathcal{M}, g, m \Vdash \neg\varphi & \text{iff } \mathcal{M}, g, m \not\Vdash \varphi \\ \mathcal{M}, g, m \Vdash \varphi_1 \wedge \varphi_2 & \text{iff } \mathcal{M}, g, m \Vdash \varphi_1 \text{ and } \mathcal{M}, g, m \Vdash \varphi_2 \\ \mathcal{M}, g, m \Vdash [R]\varphi & \text{iff } \forall m'. (R(m, m') \Rightarrow \mathcal{M}, g, m' \Vdash \varphi) \\ \mathcal{M}, g, m \Vdash @_s\varphi & \text{iff } \mathcal{M}, g, m' \Vdash \varphi, \text{ where } [V, g](s) = \{m'\}, s \in \text{SSYM} \\ \mathcal{M}, g, m \Vdash \downarrow x.\varphi & \text{iff } \mathcal{M}, g_m^x, m \Vdash \varphi. \end{array}$$

## 2.1 Direct resolution for hybrid logics

Designing resolution methods that can directly (without translation into large background languages) be applied to modal logics, received some attention in the late 1980s and early 1990s [11, 17, 10]. Also, the first (non-clausal) resolution methods for temporal languages go back to that period with the work of Abadi and Manna [1]. But even though we might sometimes think of modal languages as a “simple extension of propositional logic,” direct resolution for modal languages has proved a difficult task. Intuitively, in basic modal languages the resolution rule has to operate *inside* boxes and diamonds to achieve completeness. This leads to more complex systems, less elegant results, and poorer performance, ruining the “one-dumb-rule” spirit of resolution.

In [5] we presented a resolution calculus that used the hybrid machinery to “push formulas out of modalities” and in this way, feed them into a simple and standard resolution rule. Nominals and @ introduce a limited form of equational reasoning. A formula like @<sub>*i*</sub>*j* is true in a model iff *i* and *j* are nominals for the *same* state. In the classical resolution tradition, Robinson and Wos [19] have proposed a rule called *paramodulation* to improve previous accounts of equational reasoning in first-order logic, and we can indeed use paramodulation to handle nominals and @ in our setting.

Very briefly, our resolution algorithm works as follows. First define the following rewriting procedure *nf* on hybrid formulas:

$$\begin{aligned} \neg\neg\varphi &\stackrel{nf}{\rightsquigarrow} \varphi, \\ \langle R \rangle\varphi &\stackrel{nf}{\rightsquigarrow} \neg([R]\neg\varphi), \\ (\varphi_1 \vee \varphi_2) &\stackrel{nf}{\rightsquigarrow} \neg(\neg\varphi_1 \wedge \neg\varphi_2), \\ \neg@_t\varphi &\stackrel{nf}{\rightsquigarrow} @_t(\neg\varphi), \end{aligned}$$

Further, for any formula  $\varphi$  in  $\mathcal{H}(@, \downarrow)$ ,  $\varphi$  is satisfiable iff @<sub>*t*</sub> $\varphi$  is satisfiable, for a nominal *t* not appearing in  $\varphi$ . We define the clause set *ClSet* corresponding to  $\varphi$  to be  $ClSet(\varphi) = \{\{\@_t nf(\varphi)\}\}$ , where *t* does not appear in  $\varphi$ .

Next, let  $ClSet^*(\varphi)$  (the saturated clause set corresponding to  $\varphi$ ) be the smallest set containing  $ClSet(\varphi)$  and closed under the rules in Figure 1.

The computation of  $ClSet^*(\varphi)$  is in itself a sound and complete algorithm for checking satisfiability of  $\mathcal{H}(@)$ , in the sense that  $\varphi$  is unsatisfiable if and only if the empty clause  $\{\}$  is a member of  $ClSet^*(\varphi)$  (see [5] for details).

What about binders? Extending the system to account for hybrid sentences using  $\downarrow$  is fairly straightforward. First, extend *nf* to handle  $\downarrow$ :

$$\neg\downarrow x.\varphi \stackrel{nf}{\rightsquigarrow} \downarrow x.\neg\varphi.$$

And consider the rule ( $\downarrow$ ) below (as @,  $\downarrow$  is self dual so we don’t need a rule for its negation):

$$(\downarrow) \frac{Cl \cup \{\@_t \downarrow x.\varphi\}}{Cl \cup \{\@_t \varphi(x/t)\}}.$$

Notice that the rule transforms hybrid sentences into hybrid sentences. The full set of rules gives us a sound and complete calculus for sentences in  $\mathcal{H}(@, \downarrow)$ . Let’s go through a short example.

**Example 2.3.** We prove that  $\downarrow x.\langle R \rangle(x \wedge p) \rightarrow p$  is a tautology. Consider the clause set corresponding to the negation of the formula:

$$\begin{array}{c}
(\wedge) \frac{Cl \cup \{\@_t(\varphi_1 \wedge \varphi_2)\}}{Cl \cup \{\@_t\varphi_1\} \\ Cl \cup \{\@_t\varphi_2\}} \quad (\neg\wedge) \frac{Cl \cup \{\@_t\neg(\varphi_1 \wedge \varphi_2)\}}{Cl \cup \{\@_t\neg\varphi_1, \@_t\neg\varphi_2\}} \\
\\
(\text{RES}) \frac{Cl_1 \cup \{\@_t\varphi\} \quad Cl_2 \cup \{\@_t\neg\varphi\}}{Cl_1 \cup Cl_2} \\
\\
([R]) \frac{Cl_1 \cup \{\@_t[R]\varphi\} \quad Cl_2 \cup \{\@_t\neg[R]\neg s\}}{Cl_1 \cup Cl_2 \cup \{\@_s\varphi\}} \\
\\
(\neg[R]) \frac{Cl \cup \{\@_t\neg[R]\varphi\}}{Cl \cup \{\@_t\neg[R]\neg n\} \\ Cl \cup \{\@_n\neg\varphi\}}, \text{ where } n \text{ is new.} \\
\\
(@) \frac{Cl \cup \{\@_t\@_s\varphi\}}{Cl \cup \{\@_s\varphi\}} \\
\\
(\text{PARAM}) \frac{Cl_1 \cup \{\@_t s\} \quad Cl_2 \cup \{\varphi(t)\}}{Cl_1 \cup Cl_2 \cup \{\varphi(t/s)\}} \\
\\
(\text{SYM}) \frac{Cl \cup \{\@_t s\}}{Cl \cup \{\@_s t\}} \quad (\text{REF}) \frac{Cl \cup \{\@_t \neg t\}}{Cl}
\end{array}$$

Figure 1: Labeled resolution rules for  $\mathcal{H}(@)$ .

1.  $\{\@_i(\downarrow x.\neg[R]\neg(x\wedge p)) \wedge \neg p\}$  by  $(\wedge)$
1.  $\{\@_i\downarrow x.\neg[R]\neg(x \wedge p)\}, \{\@_i\neg p\}$  by  $(\downarrow)$
2.  $\{\@_i\neg[R]\neg(i \wedge p)\}, \{\@_i\neg p\}$  by  $(\neg[R])$
3.  $\{\@_i\neg[R]\neg j\}, \{\@_j(i\Delta p)\}, \{i:\neg p\}$  by  $(\wedge)$
4.  $\{\@_j i\}, \{\@_j p\}, \{\@_i\neg p\}$  by  $(\text{PARAM})$
5.  $\{\@_i p\}, \{\@_i\neg p\}$  by  $(\text{RES})$
6.  $\{\}$ .

### 3 The “given clause” algorithm for hybrid resolution

HyLoRes implements a version of the “given clause” algorithm (see, e.g., [20]). Figure 2 shows the general setting. The sets **NEW**, **CLAUSES**, **INUSE** and **INACTIVE** are the main repositories for clauses.

An informal description of the algorithm is as follows. After reading the input file, all formulas in the input clause set are moved to the internal representation and stored in the **NEW** set. Then, the main resolution loop, which can be described in four steps, proceeds:

1. We start by deleting from **NEW** all clauses subsumed by any formula in **CLAUSES** or **INUSE** or present in **INACTIVE**, and translating clauses to normal form. If the empty clause is found in **NEW**, the algorithm exits with “Unsatisfiable.” Otherwise, complexity values for each of the clauses are computed (length, number of variables, modal depth, etc.) The formulas are then removed from **NEW** and moved into **CLAUSES**.
2. If **CLAUSES** is found empty we have arrived at a saturation and the program exits with

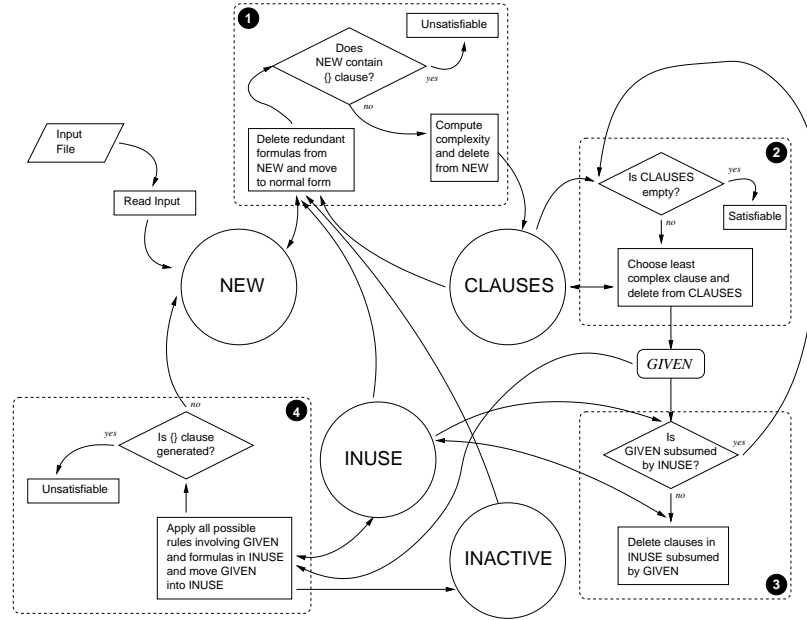


Figure 2: The “given clause” algorithm

“Satisfiable”. Otherwise, the least complex clause according to the selection criterion is chosen (and removed) from `CLAUSES` as the given clause. At the moment we use the following selection criterion: least number of formulas, then least number of variables, then least modal depth).

3. If the given clause is subsumed by formulas in `INUSE` then it is deleted and we go back to the previous step (forward subsumption). Otherwise, we proceed to delete all formulas in `INUSE` subsumed by the given clause (backwards subsumption).
4. This is the resolution step proper. All rules of inference are applied between the given clause and the clauses in `INUSE`, or just to the given clause. The resulting clauses all go into `NEW`. After all the inferences, the given clause is added to `INUSE`. If the rules for conjunction, disjunction, diamond or downarrow are applied, the given clause is also added to `INACTIVE` so that they are not generated again. If during this process the empty clause is produced, the algorithm exists with “Unsatisfiable.”

### 3.1 Some further details on the implementation

HyLoRes is implemented in Haskell, and compiled with the Glasgow Haskell Compiler Version 5.00, generating executable code which increases the usability of the prover.

We have tried our best to obtain modularity concerning, in particular, two issues: the internal representation of the different kinds of data, and the handling of new resolution rules (with the aim to extend the logical operators the prover is able to handle).

With respect to data types, we have used the fairly efficient `UnbalancedSet` type provided by the Edison package <http://www.cs.columbia.edu/~cdo/edison> to implement most of the data types representing sets. But while we represent clauses directly as `UnbalancedSet`, we have chosen different representations for each of the clause sets used by the given algorithm:

while `NEW` and `INUSE` are simply lists of clauses (as they always have to be examined linearly one by one), `CLAUSES` and `INACTIVE` are themselves `UnbalancedSets` of clauses. In particular `CLAUSES` is ordered by our selection criteria, which makes the selection of the given clause specially efficient. Furthermore, the internal state of the given clause algorithm is represented as a combination of a `State` and an `Output Monads` (see [21]). This will allow the addition of further structure (hashing functions, search tables, etc.) to optimize search, with minimum recoding.

With respect to the addition of further resolution rules, our main aim was not to disturb the modularity of the given clause algorithm. New rules can simply be added in step 4 of the main resolution loop, without the need of any further modification of the code.

As an example of the execution of the prover, we show how `HyLoRes` solves Example 2.3.

**Example 3.1.** The following are dumps of the input formula and the execution of the prover (minimally formatted for presentation).

**Input file:**

```
begin
!((down (x1 dia (x1 & p1) )) -> p1)
end
```

**Execution:**

```
(carlos@guave 149) hyllores -f test.frm -r
Input:
  {[@(NO, (-P1 & Down(X1, -[R1]-(P1 & X1))))]}
End of input

Given: (1, [@(NO, (-P1 & Down(X1, -[R1]-(P1 & X1))))])
CON: {[@(NO, -P1)] [@(NO, Down(X1, -[R1]-(P1 & X1)))]}
Given: (2, [@(NO, -P1)])
Given: (3, [@(NO, Down(X1, -[R1]-(P1 & X1)))]
ARR: {[@(NO, -[R1]-(P1 & NO))]}
Given: (4, [@(NO, -[R1]-(P1 & NO))])
DIA: {[@(N-2, (P1 & NO))] [@(NO, -[R1]-N-2)]}
Given: (5, [@(N-2, (P1 & NO))])
CON: {[@(N-2, P1)] [@(N-2, NO)]}
Given: (6, [@(N-2, NO)])
PAR (0, -2): {[@(N-2, (P1 & N-2))] [@(N-2, -[R1]-(P1 & N-2))]
 [@(N-2, Down(X1, -[R1]-(P1 & X1)))] [@(N-2, -P1)]
 [@(N-2, (-P1 & Down(X1, -[R1]-(P1 & X1)))]}
Given: (7, [@(N-2, P1)])
Given: (8, [@(N-2, -P1)])
RES: (7, [])
```

## 4 Future Work

There remain zillions of things to try and improve on `HyLoRes`. We briefly comment below on a number of things we are planning to do:

- We are starting right now with some intensive testing both with random generators like `QBF` [16], and with hand tailored examples like `Balsiger`, `Heuerding` and `Schwendimann's` [7]. At the moment we are checking the effect of different selection criteria.

- We would like to compare the performance of standard first-order resolution provers when ran on formulas of the bounded fragment, against HyLoRes running on their hybrid counterpart.
- We are planning to extend the language with the universal modality  $A$ , which will let us perform inference in terms of full Boolean knowledge bases of the description logic  $\mathcal{ALC}$ , in HyLoRes (see [2]).
- We will test and optimize the data types used.
- Some of the heuristics investigated in [6] will be implemented in HyLoRes.
- We would like to add a graphical interface for ease of use, both in the input of formulas and in the output of results. At the moment, the few parameters of HyLoRes give a very limited control on the output. We would like the prover to offer also the possibility to show a concise refutation proof in case it finds one, or to show a model if it finds a saturation.
- At the moment, the prover always attempts to use the same set of rules and heuristics, disregarding the properties of the input clause set. Making the prover aware of the characteristics of its input would lead to improvements.
- Finally, we would like to develop both the theoretical and practical issues involved in performing direct *ordered* resolution for hybrid logics.

## References

- [1] M. Abadi and Z. Manna. Nonclausal temporal deduction. *Lecture Notes in Computer Science*, 193:1–15, 1985.
- [2] C. Areces. *Logic Engineering. The Case of Description and Hybrid Logics*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands, October 2000.
- [3] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: characterization, interpolation and complexity. *The Journal of Symbolic Logic*, 66(3):977–1010, 2001.
- [4] C. Areces, H. de Nivelle, and M. de Rijke. Prefixed resolution: a resolution method for modal and description logics. In *Automated deduction—CADE-16 (Trento, 1999)*, pages 187–201. Springer, Berlin, 1999.
- [5] C. Areces, H. de Nivelle, and M. de Rijke. Resolution in modal, description and hybrid logic. *Journal of Logic and Computation*, 11(5):717–736, 2001.
- [6] Y. Auffray, P. Enjalbert, and J. Hebrard. Strategies for modal resolution: results and problems. *Journal of Automated Reasoning*, 6(1):1–38, 1990.
- [7] P. Balsiger, A. Heuerding, and S. Schwendimann. A benchmark method for the propositional modal logics  $K$ ,  $KT$ ,  $S4$ . *Journal of Automated Reasoning*, 24(3):297–317, 2000.
- [8] P. Blackburn. Internalizing labelled deduction. *Journal of Logic and Computation*, 10(1):137–168, 2000.

- [9] P. Blackburn. Representation, reasoning, and relational structures: a Hybrid Logic manifesto. In C. Areces, E. Franconi, R. Goré, M. de Rijke, and H. Schlingloff, editors, *Methods for Modalities, 1*, volume 8(3), pages 339–365. Logic Journal of the IGPL, 2000.
- [10] H. de Nivelle. *Ordering refinements of resolution*. PhD thesis, Technische Universiteit Delft, Delft. The Netherlands, October 1994.
- [11] P. Enjalbert and L. Fariñas del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65(1):1–33, 1989.
- [12] V. Haarslev and R. Möller. RACE system description. In Lambrix et al. [15], pages 130–132.
- [13] I. Horrocks. FaCT and iFaCT. In Lambrix et al. [15], pages 133–135. FACT is available under the GNU public license at <http://www.cs.man.ac.uk/~horrocks>.
- [14] U. Hustadt, R. A. Schmidt, and C. Weidenbach. MSPASS: Subsumption testing with SPASS. In Lambrix et al. [15], pages 136–137.
- [15] P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors. *Proceedings of the 1999 International Workshop on Description Logics (DL'99)*, 1999.
- [16] F. Massacci. Design and results of the Tableaux-99 Non-Classical (Modal) Systems Comparison. In N. Murray, editor, *Proceedings of the Third International Conference on Analytic Tableaux and Related Methods (TABLEAUX'99)*, volume 1617 of *Lecture Notes in Artificial Intelligence*, pages 14–18. Springer-Verlag, 1999.
- [17] G. Mints. Resolution calculi for modal logics. *American Mathematical Society Translations*, 143:1–14, 1989.
- [18] P. Patel-Schneider. DLP system description. In E. Franconi, G. De Giacomo, R. MacGregor, W. Nutt, and C. Welty, editors, *Proceedings of the 1998 International Workshop on Description Logics (DL'98)*, pages 87–89, 1998. DLP is available at <http://www.bell-labs.com/user/pfps>.
- [19] G. Robinson and L. Wos. Paramodulation and theorem-proving in first-order theories with equality. In *Machine Intelligence, 4*, pages 135–150. American Elsevier, New York, 1969.
- [20] A. Voronkov. Algorithms, datastructures, and other issues in efficient auto mated deduction. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Automated Reasoning. 1st. International Joint Conference, IJCA R 2001*, number 2083 in LNAI, pages 13–28, Siena, Italy, June 2001.
- [21] P. Wadler. Monads for functional programming. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, number 925 in LNCS. Springer Verlag, 1995.