# HyLoRes: Direct Resolution for Hybrid Logics
(System Report)

Carlos Areces  Juan Heguiabehere
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 ($i, j, k, \ldots$) and satisfiability operators ($@_i, @_j, @_k, \ldots$), increases the expressive power of the language; 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 expressivity permits also the definition of very elegant decision algorithms, where nominals and @ play, inside the object language, the role of labels, or prefixes, which are usually needed during the construction of proofs in the modal setup [6, 3]. And all these features we get with no increase in complexity: 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 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 (FOL) [1]. The more "modal" binder $\downarrow$, binding only to the *actual point of evaluation*, provides extended expressivity, but remains below the expressive power of FOL. 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 (coinciding in expressive power with the Bounded Fragment of FOL), and has interesting decidable fragments (for example, the fragment where $\downarrow$ never appears nested) [2, 1]. We refer 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 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 handling satisfiability of sentences in $\mathcal{H}(@,\downarrow)$; it implements the algorithm presented in [3]. The prototype 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, FaCT or RACER [15, 10, 9]. Contrasting it with the provers mentioned before, the most interesting feature of HyLoRes is that it is not based on tableau algorithms but on (direct) resolution. In particular, HyLoRes implements a version of the "given clause" algorithm (see, e.g. [17]), which is nowadays the standard skeleton of the most powerful FO provers. Notice also that in contrast to translation based provers like MSPASS [11], it performs resolution directly on the modal (or hybrid) input, with no translation into large background logics.

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

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

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

where $a \in \mathsf{ATOM}$, $x \in \mathsf{SVAR}$, $s \in \mathsf{SSYM}$, $R \in \mathsf{REL}$ and $\varphi, \varphi_1, \varphi_2 \in \mathsf{FORMS}$. $\mathcal{H}(@)$ 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 FOL, with $\downarrow$ as the only binding operator. A *sentence* is a formula containing no free state variables.

**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 $V : \mathsf{PROP} \cup \mathsf{NOM} \to Pow(M)$ is such that for all nominals $i \in \mathsf{NOM}$, $V(i)$ is a singleton subset of $M$. An *assignment* $g$ for $\mathcal{M}$ is a mapping $g : \mathsf{SVAR} \to M$. Given an assignment $g$, we define $g^x_m$ (an *$x$-variant of $g$*) by $g^x_m(x) = m$ and $g^x_m(y) = g(y)$ 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 new conditions defining the *satisfiability relation* are:

$$\begin{array}{rll} \mathcal{M}, g, m \Vdash a & \text{iff} & m \in [V, g](a),\ a \in \mathsf{ATOM} \\ \mathcal{M}, g, m \Vdash @_s\varphi & \text{iff} & \mathcal{M}, g, m' \Vdash \varphi, \text{ where } [V, g](s) = \{m'\},\ s \in \mathsf{SSYM} \\ \mathcal{M}, g, m \Vdash \downarrow x.\varphi & \text{iff} & \mathcal{M}, g^x_m, m \Vdash \varphi. \end{array}$$

### 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 [8, 14, 7]. 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 [3] 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 $@_i j$ is true in a model iff $i$ and $j$ are nominals for the *same* state. In the classical resolution tradition, Robinson and Wos [16] have proposed a rule called *paramodulation* to improve previous accounts of equational reasoning in FOL, 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: $nf = \{\neg\neg\varphi \Rightarrow \varphi, \langle R\rangle\varphi \Rightarrow \neg([R]\neg\varphi), (\varphi_1 \vee \varphi_2) \Rightarrow \neg(\neg\varphi_1 \wedge$

$\neg\varphi_2), \neg@_t\varphi \Rightarrow @_t(\neg\varphi)\}$. Further, for any formula $\varphi$ in $\mathcal{H}(@,\downarrow)$, $\varphi$ is satisfiable iff $@_t\varphi$ is satisfiable, for a nominal $t$ not appearing in $\varphi$. We define the clause set *ClSet* corresponding to $\varphi$ to be $\mathit{ClSet}(\varphi) = \{\{@_t nf(\varphi)\}\}$, where $t$ does not appear in $\varphi$. Next, let $\mathit{ClSet}^*(\varphi)$ (the saturated clause set corresponding to $\varphi$) be the smallest set containing $\mathit{ClSet}(\varphi)$ and closed under the rules in Figure 1.

$$(\wedge)\quad \frac{Cl \cup \{@_t(\varphi_1 \wedge \varphi_2)\}}{\begin{array}{c} Cl \cup \{@_t\varphi_1\} \\ Cl \cup \{@_t\varphi_2\}\end{array}} \qquad (\neg\wedge)\quad \frac{Cl \cup \{@_t\neg(\varphi_1 \wedge \varphi_2)\}}{Cl \cup \{@_t nf(\neg\varphi_1), @_t nf(\neg\varphi_2)\}}$$

$$(\mathrm{RES})\quad \frac{Cl_1 \cup \{@_t\varphi\} \quad Cl_2 \cup \{@_t\neg\varphi\}}{Cl_1 \cup Cl_2}$$

$$([R])\quad \frac{Cl_1 \cup \{@_t[R]\varphi\} \quad Cl_2 \cup \{@_t\neg[R]\neg s\}}{Cl_1 \cup Cl_2 \cup \{@_s\varphi\}} \qquad (\neg[R])\quad \frac{Cl \cup \{@_t\neg[R]\varphi\}}{\begin{array}{c} Cl \cup \{@_t\neg[R]\neg n\} \\ Cl \cup \{@_n nf(\neg\varphi)\}\end{array}}, \text{ for } n \text{ new.}$$

$$(@)\quad \frac{Cl \cup \{@_t@_s\varphi\}}{Cl \cup \{@_s\varphi\}}$$

$$(\mathrm{PARAM})\quad \frac{Cl_1 \cup \{@_t s\} \quad Cl_2 \cup \{\varphi(t)\}}{Cl_1 \cup Cl_2 \cup \{\varphi(t/s)\}}$$

$$(\mathrm{SYM})\quad \frac{Cl \cup \{@_t s\}}{Cl \cup \{@_s t\}} \qquad (\mathrm{REF})\quad \frac{Cl \cup \{@_t\neg t\}}{Cl}$$

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

The computation of $\mathit{ClSet}^*(\varphi)$ 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 $\mathit{ClSet}^*(\varphi)$ (see [3] for details).

What about binders? Extending the system to account for hybrid sentences using $\downarrow$ is fairly straightforward. First, extend *nf* to handle $\downarrow$: $\neg\downarrow x.\varphi \Rightarrow \downarrow x.\neg\varphi$. And consider the rule $(\downarrow)$ below (as $@$, $\downarrow$ is self dual so we don't need a rule for its negation):

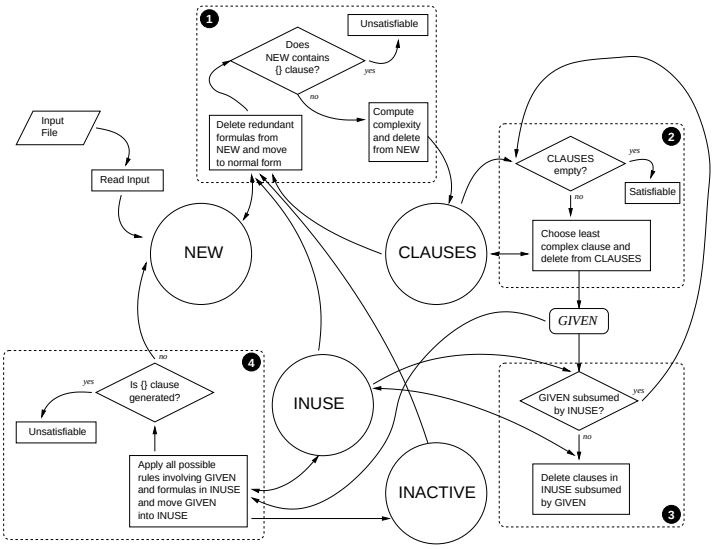
$$(\downarrow)\quad \frac{Cl \cup \{@_t\downarrow x.\varphi\}}{Cl \cup \{@_t\varphi(x/t)\}}.$$

Notice that the rule transforms hybrid sentences into hybrid sentences. The full set of rules gives us a sound and complete calculus for checking satisfiability of sentences in $\mathcal{H}(@,\downarrow)$.

**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:

1. $\{@_i((\downarrow x.\neg[R]\neg(x \underline{\wedge} p)) \wedge \neg p)\}$ by $(\wedge)$
2. $\{@_i\underline{\downarrow x}.\neg[R]\neg(x \wedge p)\}, \{@_i\neg p\}$ by $(\downarrow)$
3. $\{@_i\underline{\neg[R]}\neg(i \wedge p)\}, \{@_i\neg p\}$ by $(\neg[R])$
4. $\{@_i\underline{\neg[R]}\neg j\}, \{@_j(i \underline{\wedge} p)\}, \{i{:}\neg p\}$ by $(\wedge)$
5. $\{@_j\underline{i}\}, \{@_j\underline{p}\}, \{@_i\neg p\}$ by (PARAM)
6. $\{@_i\underline{p}\}, \{@_i\underline{\neg p}\}$ by (RES)
7. $\{\}$.

# 3 The "given clause" algorithm for hybrid resolution




HyLoRes implements a version of the "given clause" algorithm (see, e.g., [17]). The figure 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 "Satisfiable." Otherwise, the least complex clause according to the selection criterion is chosen (and removed) from CLAUSES as the given clause.
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 [18]). This will allow the addition of further structure (hashing functions, 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) hylores -f test.frm -r
Input:
   {[@(N0, (-P1 & Down(X1, -[R1]-(P1 & X1))))]}
End of input

Given: (1, [@(N0, (-P1 & Down(X1, -[R1]-(P1 & X1))))])
CON: {[@(N0, -P1)][@(N0, Down(X1, -[R1]-(P1 & X1)))]}
Given: (2, [@(N0, -P1)])
Given: (3, [@(N0, Down(X1, -[R1]-(P1 & X1)))])
ARR: {[@(N0, -[R1]-(P1 & N0))]}
Given: (4, [@(N0, -[R1]-(P1 & N0))])
DIA: {[@(N-2, (P1 & N0))][@(N0, -[R1]-N-2)]}
Given: (5, [@(N-2, (P1 & N0))])
CON: {[@(N-2, P1)][@(N-2, N0)]}
Given: (6, [@(N-2, N0)])
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 [13], and with hand tailored examples like Balsiger, Heuerding and Schwendimann's [5]. At the moment we are checking the effect of different selection criteria.
- We would like to compare the performance of standard FO resolution provers when run on formulas of the bounded fragment, against HyLoRes running on their hybrid counterpart.

- We plan 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 [1]).
- We will test and optimize the data types used.
- Some of the heuristics investigated in [4] 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] C. Areces. *Logic Engineering. The Case of Description and Hybrid Logics.* PhD thesis, ILLC, University of Amsterdam, Amsterdam, The Netherlands, October 2000.

[2] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: Characterization, interpolation and complexity. *J. of Symbolic Logic*, 66(3):977–1010, 2001.

[3] C. Areces, H. de Nivelle, and M. de Rijke. Resolution in modal, description and hybrid logic. *J. of Logic and Computation*, 11(5):717–736, 2001.

[4] Y. Auffray, P. Enjalbert, and J. Hebrard. Strategies for modal resolution: results and problems. *J. of Automated Reasoning*, 6(1):1–38, 1990.

[5] P. Balsiger, A. Heuerding, and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. *J. of Automated Reasoning*, 24(3):297–317, 2000.

[6] P. Blackburn. Internalizing labelled deduction. *J. of Logic and Computation*, 10(1):137–168, 2000.

[7] H. de Nivelle. *Ordering refinements of resolution.* PhD thesis, Technische Universiteit Delft, Delft. The Netherlands, October 1994.

[8] P. Enjalbert and L. Fariñas del Cerro. Modal resolution in clausal form. *Theoretical Computer Science*, 65(1):1–33, 1989.

[9] V. Haarslev and R. Möller. RACE system description. In Lambrix et al. [12], pages 130–132.

[10] I. Horrocks. FaCT and iFaCT. In Lambrix et al. [12], pages 133–135.

[11] U. Hustadt, R. A. Schmidt, and C. Weidenbach. MSPASS: Subsumption testing with SPASS. In Lambrix et al. [12], pages 136–137.

[12] P. Lambrix, A. Borgida, R. Möller, M. Lenzerini, and P. Patel-Schneider (eds). *Proc. of DL'99*, 1999.

[13] F Massacci. Design and results of the tableaux-99 non-classical (modal) systems comparison. In N. Murray (ed), *Proc. of TABLEAUX'99*, number 1617 of LNAI, pages 14–18. Springer, 1999.

[14] G. Mints. Resolution calculi for modal logics. *American Mathematical Society Translations*, 143:1–14, 1989.

[15] P. Patel-Schneider. DLP system description. In E. Franconi, G. De Giacomo, R. MacGregor, W. Nutt, and C. Welty (eds), *Proc. of DL'98*, pages 87–89, 1998.

[16] 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.

[17] A. Voronkov. Algorithms, datastructures, and other issues in efficient automated deduction. In R.Goré, A. Leitsch, and T. Nipkow (eds), *Automated Reasoning. 1st. International Joint Conference, IJCAR 2001*, number 2083 in LNAI, pages 13–28, Siena, Italy, June 2001.

[18] P. Wadler. Monads for functional programming. In J. Jeuring and E. Meijer (eds), *Advanced Functional Programming*, number 925 in LNCS. Springer, 1995.