HyLoRes 1.0: Direct Resolution for Hybrid Logics

Carlos Areces and Juan Heguiabehere

Language and Inference Technology
Faculty of Science. University of Amsterdam
{carlos,juanh}@science.uva.nl

Hybrid languages are modal languages that allow direct reference to the elements of a model. Even the basic hybrid language ($\mathsf{H}(@)$), which only 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: it can explicitly check whether the point of evaluation is a specific, named point in the model ($w \Vdash i$), and whether a named point satisfies a given formula ($w \Vdash @_i\varphi$). The extended expressivity allows one to define elegant decision algorithms, where nominals and @ play the role of labels, or prefixes, which are usually needed during the construction of proofs in the modal setup [5, 3]. Note that they do so *inside* the object language. All these features we get with no increase in complexity: the complexity of the satisfiability problem for $\mathsf{H}(@)$ is the same as for the basic modal language, PSPACE [2]. When we move to very expressive hybrid languages containing binders, we obtain an impressive boost in expressivity, but usually we also move beyond the boundaries of decidability. Classical binders like and (together with @) make the logic as expressive as first-order logic (FOL) while adding the more "modal" binder $\downarrow$ gives a logic weaker than FOL [1]. We refer to the Hybrid Logic site at `http://www.hylo.net` for a broad on-line bibliography.

HyLoRes is a direct resolution prover for hybrid logics handling satisfiability of sentences in $\mathsf{H}(@, \downarrow)$; it implements the algorithm presented in [3]. The most interesting distinguishing feature of HyLoRes is that it is not based on tableau algorithms but on (direct) resolution. HyLoRes implements a version of the "given clause" algorithm, which has become the skeleton underlying most first-order provers. In contrast to translation based provers like MSPASS [8], HyLoRes performs resolution directly on the modal (or hybrid) input, with no translation into background logics. It is often said that hybrid logics combine interesting features from both modal and first-order logics. In the same spirit, HyLoRes fuses ideas from state-of-the-art first-order proving with the simple representation of the hybrid object language.

HyLoRes (and the Tcl/Tk interface xHyLoRes) is available for on-line use and downloading at `http://www.illc.uva.nl/~carlos/HyLoRes`.

1. Direct resolution for hybrid logics. In [3] we presented a resolution calculus that uses 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. A paramodulation rule similar to the one used by Robinson and Wos [12] lets us handle nominals and @.

Very briefly, our resolution algorithm works as follows. First define the following rewriting procedure *nf* on hybrid formulas: $nf = \{\!\{\neg\neg\varphi \quad \varphi,\ R\ \varphi \quad \neg([R]\neg\varphi), (\varphi_1 \quad \varphi_2) \quad \neg(\neg\varphi_1 \quad \neg\varphi_2), \neg@_t\varphi \quad @_t(\neg\varphi)\}\!\}$. Further, for any formula φ in $\mathsf{H}(@,\downarrow)$, φ is satisfiable iff $@_t\varphi$ is satisfiable, for a nominal t not appearing in φ. We define the clause set *ClSet* corresponding to φ to be $\mathit{ClSet}(\varphi) = \{\!\{@_t nf(\varphi)\}\!\}$, where t does not appear in φ. Next, let $\mathit{ClSet}^*(\varphi)$ (the saturated clause set corresponding to φ) be the smallest set containing $\mathit{ClSet}(\varphi)$ and closed under the following rules.

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

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

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

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

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

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

The computation of $\mathit{ClSet}^*(\varphi)$ is in itself a sound and complete algorithm for checking satisfiability of $\mathsf{H}(@)$, in the sense that φ is unsatisfiable if and only if the empty clause $\{\!\}\!\}$ is a member of $\mathit{ClSet}^*(\varphi)$ (see [3]).

The hybrid binder $\downarrow$ binds variables to the point of evaluation, i.e., for a model M, an assignment g and a state w, $\mathsf{M}, g, w \models \downarrow x.\varphi$ iff $\mathsf{M}, g^x_w, w \models \varphi$, where g^x_w is the assignment that coincides with g, but maps x to w. For example, a state w satisfies the formula $\downarrow x.\Diamond x$ if and only if w can reach itself through the accessibility relation.

Extending the system to account for hybrid sentences using ↓ is fairly straightforward. First, extend *nf* to handle ↓: $\neg\downarrow x.\varphi \qquad \downarrow x.\neg\varphi$. Then consider the rule (↓) below

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

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

2. The "given clause" algorithm for hybrid resolution. HyLoRes implements a version of the "given clause" algorithm (see, e.g., [13]).

```
input: init: set of clauses
var: new, clauses, inuse, inactive: set of clauses
var: given: clause

clauses := {}
new := init
simplify(&new, inuse ∪ inactive ∪ clauses)
if {} ∈ new then return "unsatisfiable"
clauses := computeComplexity(new)
while clauses ≠ {} do
    given := select(clauses)
    clauses := clauses – {given}
    while subsumed(given, inuse) do
        if clauses = {}
            then return "satisfiable"
            else
                given := select(clauses)
                clauses := clauses – {given}
    simplify(&inuse, given)
    new := infer(inuse, given, &inactive)
    simplify(&new, inuse ∪ inactive ∪ clauses)
    if {} ∈ new then return "unsatisfiable"
    clauses := clauses ∪ computeComplexity(new)
```

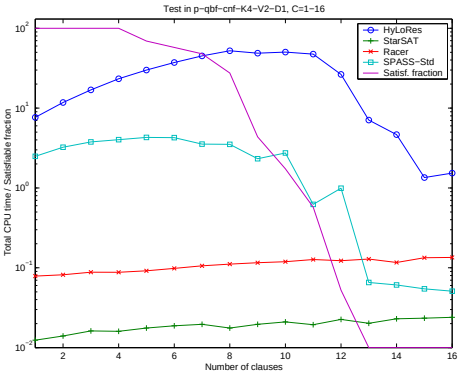
simplify performs subsumption deletion (& marks the modified set). *computeComplexity* determines length, modal depth, number of literals, etc. for each of the formulas; these values are used by *select*. *infer* applies the resolution rules to the given clause and each clause in inuse, if the ∧, ∨, ◇ or ↓-rules are applied, the given clause is added to inactive so that it's not generated again.

The implementation preserves the soundness and completeness of the calculus introduced in § 1, and ensures termination for $\mathsf{H}(@)$.

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

The design of the algorithm is modular, both in the internal representation of the different kinds of data, and in the handling of new resolution rules (so that the prover can easily be made to handle new logical operators). We have used the Edison package (a library of efficient data types provided with GHC) 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 algorithm: NEW and INUSE are simply lists of clauses (as they always have to be examined linearly one by one), CLAUSES and INACTIVE are UnbalancedSets of clauses.[1] In particular, CLAUSES is ordered by our selection criterion, which makes for an efficient selection of the given clause. The internal state of the given clause algorithm is represented as a combination of a state and an output monad. This allows 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 the *infer* function without the need for any further modification of the code.

3. Testing and Comparison. The prototype is not yet meant to be competitive when compared with state of the art provers for modal and description logics like DLP, FaCT, MSPASS or RACER [11, 7, 8, 6].



On the one hand, the system is still in a preliminary stage of development (only very simple optimizations for hybrid logics have been developed and implemented), and on the other hand the hybrid and description languages are related but different. $H(@, \downarrow)$ is undecidable while the implemented description languages are mostly decidable. And even when comparing the fragment $H(@)$ for which HyLoRes implements a decision algorithm, the expressive powers are incomparable ($H(@)$ permits free Boolean combinations of @ and nominals but lacks the limited form of universal modality available in the T-Box of DL provers [1]). The plot shows some ongoing work on basic testing with the random QBF generator [10].

[1] While `List` provides efficient sequential access to their elements, `UnbalancedSet` implements sets as unbalanced search trees to optimize search of single elements.

4. Future Work. There remain many things to try and improve in HyLoRes: 1) Develop both the theoretical and practical issues involved in performing direct *ordered* resolution for hybrid logics (ongoing). 2) Test and optimize the data types used (ongoing). 3) Make the prover aware of the characteristics of its input. At the moment, the prover always attempts to use the same set of rules and heuristics, disregarding syntactic properties of the input clause set (ongoing). 4) 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]). 5) Implement some of the heuristics presented in [4]. 6) Display a concise refutation proof in case it finds one, or a model otherwise.

As we said in the introduction, HyLoRes fuses nicely some ideas from state-of-the-art first-order proving with the simplicity of hybrid languages; and it provides the basis for future developments on computational tools for hybrid logic. Already in its actual state, users find the tool useful for better understanding the formalisms.

Acknowledgement. C. Areces was supported by the NWO project # 612.069.006.

References

1. C. Areces. *Logic Engineering. The Case of Description and Hybrid Logics*. PhD thesis, ILLC, University of Amsterdam, The Netherlands, October 2000.
2. C. Areces, P. Blackburn, and M. Marx. Hybrid logic: Characterization, interpolation and complexity. *J. Symb. Logic*, 66(3):977–1010, 2001
3. C. Areces, H. de Nivelle, and M. de Rijke. Resolution in modal, description and hybrid logic. *J. Logic and Comp.*, 11(5):717–736, 2001.
4. Y. Auffray, P. Enjalbert, and J. Hebrard. Strategies for modal resolution: results and problems. *J. Autom. Reas.*, 6(1):1–38, 1990.
5. P. Blackburn. Internalizing labelled deduction. *J. Logic and Comp.*, 10(1):137–168, 2000.
6. V. Haarslev and R. Möller. RACE system description. In Lambrix et al. [9], pages 130–132.
7. I. Horrocks. FaCT and iFaCT. In Lambrix et al. [9], pages 133–135.
8. U. Hustadt, R. A. Schmidt, and C. Weidenbach. MSPASS: Subsumption testing with SPASS. In Lambrix et al. [9], pages 136–137.
9. P. Lambrix, A. Borgida, R. Möller M. Lenzerini, and P. Patel-Schneider, (eds.) *Proc. of the 1999 Int. Workshop on Description Logics (DL'99)*, 1999.
10. 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.
11. P. Patel-Schneider. DLP system description. In E. Franconi, G. De Giacomo, R. MacGregor, W. Nutt, and C. Welty, (eds.), *Proc. of the 1998 Int. Workshop on Description Logics (DL'98)*, pages 87–89, 1998.
12. G. Robinson and L. Wos. Paramodulation and theorem-proving in first-order theories with equality. In *Mach. Int.* 4:135–150, 1969.
13. A. Voronkov. Algorithms, datastructures, and other issues in efficient automated deduction. In R. Goré, A. Leitsch, and T. Nipkow, (eds.), *Automated Reasoning. 1st. Int. Joint Conf., IJCAR 2001*, number 2083 of LNAI, pages 13–28, Italy, 2001.