Experiments in Theorem Proving for Topological Hybrid Logic

Dmitry Sustretov, Guillaume Hoffmann, Carlos Areces, Patrick Blackburn

HAL Id: inria-00187303
https://hal.inria.fr/inria-00187303

Submitted on 14 Nov 2007

Experiments in Theorem Proving for Topological Hybrid Logic

Dmitry Sustretov, Guillaume Hoffmann, Carlos Areces, and Patrick Blackburn

TALARIS
INRIA Lorraine
54602 Villers-lès-Nancy, France
firstname.lastname@loria.fr

Abstract

This paper discusses two experiments in theorem proving for hybrid logic under the topological interpretation. We begin by discussing the topological interpretation of hybrid logic and noting what it adds to the topological interpretation of orthodox modal logic. We then examine two implemented proof methods. The first makes use of HyLoBan, a terminating theorem prover that searches for a winning search strategy in certain topologically motivated games. The second is a translation-based approach that makes use of HyLoTab [18], a tableaux-based theorem prover for hybrid logic under the standard relational interpretation. We compare the two methods, and note a number of directions for further work.

Keywords: hybrid logic, topological semantics, theorem proving

1 Introduction

Topological semantics for modal logic is 20 years older than the (now standard) relational semantics; moreover, it was the first framework in which deep technical results about modal logic were proved. Alfred Tarksi's 1938 paper [17] defined the semantics and showed that **S4** is complete with respect to the class of all topological spaces. Then, in 1944, McKinsey and Tarski [12] proved an elegant result: **S4** is also the modal logic of the real numbers under the usual topology.

After the birth of relational semantics in the 1960s, topological semantics was somewhat neglected, though technically interesting results continued to be proved (see for example Esakia [6] and Shehtman [14]). More recently, however, partly because of the growing interest in logics of space and in developing topological accounts of knowledge, there has been a revival of interest; a good illustration of such work is Aiello, van Benthem, and Bezhanishvili [1]. A theme emphasized in this newer phase (particularly by van Benthem and his various co-authors) is the need to move beyond the basic "box and diamond" modal language. As they point out, the basic language is highly inexpressive with respect to the kinds of topological

spaces of interest to mathematicians. Indeed, this is already clear from the classic McKinsey and Tarski results mentioned above. The real numbers (under the usual topology) satisfy what topologists call the T_0 and T_1 separation axioms, and much else besides; that is, the reals are a space with many special topological properties. However the basic modal language sees no difference between this space and the class of all topological spaces; both share the same modal logic, namely **S4**.

Recent work shows that matters become rather more interesting when the basic modal language is enriched with the basic tools of hybrid logic, namely nominals and the universal modality. As we shall discuss below, the increased expressivity means that the hybrid logics of all topological spaces, of T_0 topological spaces, and of T_1 topological spaces are all distinct.

But this increased expressivity has consequences for theorem proving. Nothing needs to be said about topological theorem proving for ordinary modal logic — it's plain old **S4** theorem proving, thus many good solutions already exist. But theorem proving for hybrid logic is less well developed. In particular, at the moment there are no terminating provers which handle logics richer than the minimal hybrid logic **K**, or which cope with the universal modality. Now, our goal is to incorporate topological theorem proving within the InToHyLo suit (Inference Tools for Hybrid Logic; see [2]), a general inference framework for hybrid logic. Doing so will require efficient and terminating tools for handling an **S4** modality, nominals, the universal modality, and topologically motivated constraints. Here we report on two preliminary experiments which we believe point the way to such an implementation. Both approaches are based on Sustretov's reductions of the hybrid logics of T_0 and T_1 spaces to the *relational* hybrid logics of two classes of finite frames (see [15,16]). But the two approaches exploit Sustretov's reductions differently:

- First we discuss a game-based prover called HyLoBan, a direct implementation of Sustretov's game-based proofs of the PSPACE-completeness of the logics of T_0 and T_1 spaces. The interest of this approach is that termination is guaranteed and the underlying game-based architecture seems of independent interest; its disadvantage is that (at present) it is extremely inefficient.

- The second approach makes use of the fact that the relevant classes of finite frames used in the reductions can be encoded with the help of the universal modality. Thus we can translate topological satisfiability problems into relational satisfiability problems involving the universal modality, and solve them using HyLoTab [18], the only existing prover capable of handling an **S4** modality together with the universal modality and nominals. This approach turns out to be more efficient than the present implementation of HyLoBan; its disadvantage is that HyLoTab is not an optimised prover and is not guaranteed to terminate on all inputs (and indeed, as we shall see, it can loop on quite simple formulas).

We proceed as follows. In Section 2 we discuss topological semantics for hybrid logic, and the hybrid axiomatisations of T_0 and T_1 spaces. In Section 3 we present the game-based approach to topological theorem proving, and its implementation in HyLoBan. In Section 4 we discuss the translation-based approach using HyLoTab. In Section 5 we evaluate the two approaches, and in Section 6 we conclude.

2 Topological Semantics for Hybrid Logics

We assume that the reader is familiar with the basics of hybrid logic under the relational interpretation (for example, [4] contains all the required background). Here we are going to work with the basic hybrid language, but under another semantics: formulas will be interpreted on topological spaces.

The language we shall work with is generated by the following grammar:

$$\varphi ::= p \mid i \mid \varphi \wedge \varphi \mid \neg\varphi \mid \Box\varphi \mid E\varphi$$

where p is one of the ordinary propositional letters and i is one of the distinguished propositional letters called nominals. We use letters $p, q, r, \ldots$ for ordinary propositional variables and $i, j, k, \ldots$ for nominals. We define dual modalities $\Diamond$ and A as usual: $\Diamond\varphi \equiv \neg\Box\neg\varphi$ and $A\varphi \equiv \neg E\neg\varphi$, and we sometimes write $@_i\varphi$ for $E(i \wedge \varphi)$. Nominals are required to always evaluate to singleton sets and $E\varphi$ is interpreted as "there exists some point in the model where φ holds". This interpretation of the hybrid machinery is quite general and has been most often used together with classical relational interpretation of the modal operators. As we shall now see, these ideas transfer straightforwardly to the topological treatment of modality.

Definition 2.1 (Topological models) *A topological space is a pair (T, τ) where $\tau \subseteq \mathcal{P}(T)$ such that $\emptyset, T \in \tau$ and τ is closed under finite intersections and arbitrary unions. Elements of τ are called* open sets *or* opens, *and an open containing a point x is called a* neighborhood *of the point x. Complements of open sets are called* closed sets.

A topological model $\mathfrak{M}$ is a tuple (T, τ, V) where (T, τ) is a topological space and the valuation V maps propositional letters and nominals to subsets of T, with nominals always being assigned singleton subsets.

Definition 2.2 (Topological semantics) *Truth of a formula φ at a point w in a topological model $\mathfrak{M}$ (denoted by $\mathfrak{M}, w \models \varphi$) is defined inductively as follows:*

$$
\begin{aligned}
&\mathfrak{M}, w \models p && \textit{iff} \quad x \in V(p) \\
&\mathfrak{M}, w \models i && \textit{iff} \quad x \in V(i) \\
&\mathfrak{M}, w \models \varphi \wedge \psi && \textit{iff} \quad \mathfrak{M}, w \models \varphi \textit{ and } \mathfrak{M}, w \models \psi \\
&\mathfrak{M}, w \models \neg\varphi && \textit{iff} \quad \mathfrak{M}, w \nvDash \varphi \\
&\mathfrak{M}, w \models \Box\varphi && \textit{iff} \quad \exists O \in \tau \textit{ such that } w \in O \textit{ and } \forall v \in O.(\mathfrak{M}, v \models \varphi) \\
&\mathfrak{M}, w \models E\varphi && \textit{iff} \quad \exists v \textit{ such that } \mathfrak{M}, v \models \varphi.
\end{aligned}
$$

It follows that for all nominals i, $\mathfrak{M}, w \models @_i\varphi$ iff there is v such that $\mathfrak{M}, v \models i$ and $\mathfrak{M}, v \models \varphi$, just as in relational semantics.

What is known about hybrid logic under this interpretation? For a start, the hybrid logic of all topological spaces coincides with the hybrid logic of transitive reflexive frames under the relational semantics: that is, both are hybrid **S4**. However it's not "**S4** all the way up to the Reals," as is the case for orthodox modal languages. It turns out that the hybrid machinery is sensitive to the two simplest *separation axioms*, the conditions that define what topologists call T_0 and T_1 spaces:

Definition 2.3 (Separation axioms)

T_0 *for any two distinct points x, y there is either an open neighborhood of x that does not contain y, or an open neighborhood of y that does not contain x;*

T_1 *any singleton set is closed.*

Both conditions are definable in the hybrid language. The formulas

$$(T_0)\ @_i\neg j \to (@_i\square\neg j \vee @_j\square\neg i)\ \text{ and }\ (T_1)\ \Diamond i \to i$$

define the classes of T_0 and T_1 spaces respectively. We denote the hybrid logics of these spaces as $\mathsf{Log}(T_0)$ and $\mathsf{Log}(T_1)$. It is easy to see that every T_1 space is T_0 space (but not conversely) and hence $\mathsf{Log}(T_0)$ is a proper subset of $\mathsf{Log}(T_1)$.

Let's take a closer look at these axioms, starting with the simpler T_1 axiom. This may be familiar under its *relational* interpretation: there $\Diamond i \to i$ defines the class of frames that consist of isolated reflexive points. The hybrid logic of this class of frames is barely different from classical propositional logic and is NP-complete. On the other hand, as we've just said, in topological semantics this axiom defines the class of T_1 spaces, whose logic is far richer — in fact, it is PSPACE-complete (see [15,16] for details). As this example makes clear, the same axiom may have quite different effects in the two semantics, and these differences can affect both the proof theory and the computational complexity of the resulting logics. Similarly, the more complex formula defining T_0 spaces has a very different meaning in relational semantics: there it defines the class of antisymmetric frames.

In spite of these differences, it is possible to characterise the topological logics $\mathsf{Log}(T_0)$ and $\mathsf{Log}(T_1)$ in relational terms, and indeed all our subsequent work depends on this reduction. In particular, Sustretov [15,16] has proved that these logics are complete with respect to classes of finite transitive and reflexive *relational* models satisfying some extra condition. Those conditions are:

Definition 2.4 (Relational model conditions)

T_0 *There are no non-trivial cycles involving points named by nominals;*

T_1 *Points named by nominals have no incoming arcs other than from themselves.*

[15,16] uses this reduction to show that $\mathsf{Log}(T_0)$ and $\mathsf{Log}(T_1)$ are both PSPACE-complete. These are the logics on which we will conduct our first topological theorem proving experiments. We will investigate two approaches, both of which depend on this relational characterisation. In the first experiment, we will directly implement Sustretov's PSPACE algorithm. In the second, we shall characterise the frame classes just mentioned with the help of the universal modality, and then hand the universal-modality-encoded-problem to a tableau-based prover.

3 Game-based Satisfiability Checking

In this section we introduce $\mathsf{HyLoBan}$[1], a proof of concept implementation of the game-based approach developed in [15,16].

---

[1] The name is an allusion to Sokoban, a game that is recently proven to be PSPACE-complete. Our prover plays games in order to do its job; hence, since the hybrid logics it deals with are PSPACE-complete, one can (in theory!) use our prover to play Sokoban (or conversely, use Sokoban to judge topological satisfiability).

The prover works by searching for a winning strategy in a two player game which we will present below; there are two variants of the game: one for T_0, another for T_1. The game is played by putting structures called Hintikka sets on the board and linking them with each other by a relation.

Definition 3.1 (Hintikka set) *Let Σ be a set of formulas closed under subformulas and single negations (from now on, we will denote the closure of a set of formulas Γ under subformulas and single negations as $Cl(\Gamma)$). A set $A \subseteq \Sigma$ is called a* Hintikka set *if it is a maximal subset satisfying the following conditions:*

(i) *if $\neg\varphi \in \Sigma$ then $\varphi \in A$ iff $\neg\varphi \notin A$*

(ii) *if $\varphi \wedge \psi \in \Sigma$ then $\varphi \wedge \psi \in A$ iff $\varphi \in A$ and $\psi \in A$.*

There are two players: ∀belard (male) and ∃loise (female). Let φ be the formula that they are checking for satisfiability. ∃loise plays by putting Hintikka sets on the board and defining a transitive and reflexive relation R on them; ∀belard introduces challenges that she must meet. ∃loise starts the game by putting a set $\{X_0, \dots, X_k\}$ (for $k \leq |Cl(\varphi)|$) on the board, and defining R as the minimal reflexive relation on them. The sets must satisfy the following conditions:

(ROOT) X_0 contains φ,

(INIT-NOM)   each nominal appears in exactly one Hintikka set,

(INIT-UNIV)   for all X_l and all $E\chi \in Cl(\varphi)$, $E\chi \in X_l$ iff $\chi \in X_j$ for some j,

(INIT-DIAMOND) for all $\Diamond\chi \in Cl(\varphi)$, if RX_lX_j and $\Diamond\chi \notin X_l$ then $\Diamond\chi \notin X_j$ and $\chi \notin X_j$,

If the conditions do not hold, ∃loise loses immediately. ∀belard's turn consists of selecting a Hintikka set X_l and picking a formula $\Diamond\psi$ out of it. ∃loise must meet the challenge by putting a Hintikka set Y on the board and link it with X_l, such that the following conditions hold:

(DIAMOND) $\psi \in Y$, RX_lY and for all $\Diamond\chi \in Cl(\varphi)$, if $\Diamond\chi \notin X_l$ then $\Diamond\chi \notin Y$ and $\chi \notin Y$,

(UNIV)   for all X_l and for all $E\chi \in Cl(\varphi)$, $E\chi \in X_l$ iff $\chi \in X_j$ for some j,

(NOM)   if $i \in Y$ for some nominal i then Y is one of the Hintikka sets ∃loise played during the first move. If this is the case, the game stops and she wins (unless one of the next two special rules is violated, in which case she loses),

(CYCLES) R does not have non-trivial cycles that involve Hintikka sets that contain distinct nominals [**for the T_0 game**].

(NO-INCOMING) points named by nominals have no incoming arcs other than from themselves [**for the T_1 game**],

If ∃loise cannot find a Y that satisfies those conditions, then the game stops and ∀belard wins. Otherwise, ∀belard must choose a formula of the form $\Diamond\psi$ from the last played set (that is, Y) and the game continues in a similar way. If ∃loise manages to meet all ∀belard's challenges and if he has no more challenges to present,

she wins. This does not guarantee that the game will stop at some point, so we introduce an extra rule. A list of formulas played by ∀belard is kept, if he plays a formula a second time, ∃loise must respond with the same Hintikka set as she did when he played the formula for the first time. If her set satisfies the conditions from the previous paragraph, ∃loise wins; otherwise, she loses. In any case, the game stops immediately.

3.1 Implementing the game

HyLoBan is written in the functional language Haskell [11], using the Glasgow Haskell Compiler (GHC) [8]. The code is released under the GNU GPL and can be downloaded from http://hylo.loria.fr/intohylo/hyloban.php.

Apart from the main loop of the algorithm, which is an instance of minimax, the most important part of the implementation is the generation of Hintikka sets. At each turn, ∃loise plays Hintikka sets subject to certain conditions on the board. This means that the implementation should include an efficient procedure for generating Hintikka sets that satisfy given conditions.

Our current implementation generates all possible Hintikka sets from the input formula at the beginning of the game. In the course of the game when we need Hintikka sets that meet particular conditions, we scan the generated Hintikka sets and filter the good ones. Let us see how this is done.

3.2 How ∃loise moves

∃loise's first turn: For her first move, ∃loise's natural strategy is to put as few Hintikka sets as possible on the board in order to reduce the chances of ∀belard finding a challenge that will make her lose. Therefore, our implementation tries to generate initial boards as small as possible.

The conditions that must be fulfilled by the Hintikka sets that are put on the board during the first turn are the following:

- at least one formula must contain the input formula φ,
- every nominal which occurs in the input formula should belong to some set.

For each formula $E\psi \in Cl(\varphi)$, the (INIT-UNIV) condition leaves two possibilities which lead to further constraints:

- ψ belongs to one of the Hintikka sets and $E\psi$ should belong to all generated Hintikka sets (let us say then that ψ *occurs existentially*),
- ψ and $E\psi$ should not belong to any of the generated Hintikka sets ($\neg\psi$ *occurs universally*).

Note that some conditions have an impact on all generated Hintikka sets while some only concern individual Hintikka sets. If we want to generate all possible Hintikka sets, we should consider all combinations of conditions of the second type.

Since every condition should be satisfied by at least one Hintikka set, it seems plausible to use the following approach. We generate all possible partitions of the set of all conditions. Each equivalence class of a partition corresponds to a Hintikka set that satisfies conditions from this class.

6

For example, consider the formula $\varphi = i \vee j$. We have three conditions associated with this formula: i should occur somewhere, j should occur somewhere, φ should occur somewhere. Possible partitions are:

$$i \mid j \mid \varphi \qquad\qquad i, j \mid \varphi \qquad\qquad i \mid j, \varphi$$
$$i, \varphi \mid j \qquad\qquad i, j, \varphi$$

In our implementation we generate all partitions of conditions using the technique described in [13].

For each generated partition, we go through its equivalence classes and for each of them we generate all Hintikka sets that satisfy the conditions in that class. We then put together Hintikka sets that satisfy sets of conditions from different equivalence classes to form candidate initial boards. Then for every generated initial board, all the "global" conditions (for example, that there is no nominal that belongs to several distinct Hintikka sets) are checked in order to ensure that it is well-formed.

Existential formulas are treated separately. Before the generation of partitions we go through all formulas of the form $E\psi$ from $Cl(\varphi)$ and decide for each of them if ψ should occur existentially or $\neg\psi$ should occur universally. In the first case we get one "individual" condition that participates in partition generation and a "global" condition, while in the other case we have two global conditions. We then generate the partitions and initial boards as described above. This procedure is repeated for all possible combinations of occurrence types of ψs.

∃loise's subsequent turns: When ∀belard points to a formula $\diamond\psi$ on the board, the Hintikka set that ∃loise builds in response must contain ψ. Moreover, it must not contain any ψ for which $E\psi \in Cl(\varphi)$ and there is already in the board a Hintikka set that does not contain $E\psi$.

When ∀belard reuses a formula, ∃loise must answer with the same Hintikka set that she used to respond to the formula the first time. In such cases, there is no Hintikka set to be generated. Hence we keep a map between formulas put on the board by ∀belard, and the Hintikka sets used to respond to them by ∃loise; we use this information to retrieve the required previously-played Hintikka set.

3.3 Structures

Hintikka sets: We represent the set of all possible Hintikka sets of the input formula as a binary tree: each branch represents a set (see the example in Figure 1). A node at distance n from the root of the tree represents the nth formula in the list of all positive formulas of $Cl(\varphi)$, and for each node, the left (resp. right) outgoing edge represents the choice of including this formula (resp. its negation) in the set. A leaf that is not at distance $n + 1$ from the root means that there is no possible set with the choices made in its branch.

Let $c = |Cl(\varphi)|$. With a simple list of all sets, the maximum size needed would be $c * 2^c$, whereas the binary tree needs at most 2^{c+1} nodes. So the binary tree provides a smaller representation, and hence faster Hintikka sets queries.

The Board: HyLoBan uses a global state where the main data structure is *Board-Data*, which contains a *Board* object, the non-negative subformula closure of the input formula and the set of all possible Hintikka sets for the input formula. The
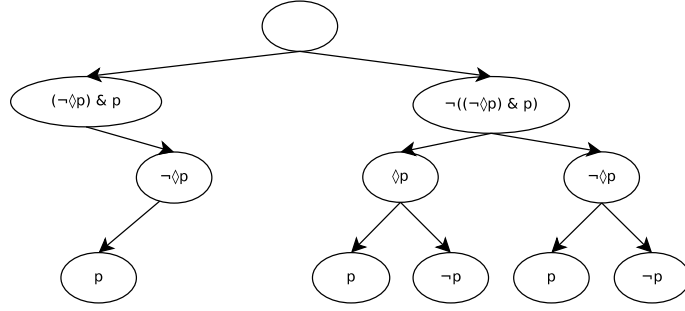
Figure 1. Representation of the possible Hintikka sets of $\neg(\Diamond p) \wedge p$ as a tree

main components of the Board object are three lists and a matrix:

- *hSets :: [HintikkaSet]*. The Hintikka sets on the board. The order in the list matters: the tail of the list is the latest Hintikka set added.
- *relationMatrix :: Matrix*. Represents the R relation between Hintikka sets. We do not enforce the reflexivity and transitivity of R, but we extend the (DIAMOND) condition to check reflexive consistency.
- *firstHSets :: [Int]*. Ordered indexes (among all possible Hintikka sets for the input formula) of the first Hintikka sets put on the board. This serves as a hash for the board.
- *forcedFormulas :: [Formula]*. Formulas that must be present in all Hintikka sets. For each formula $E(\psi)$ in $Cl(\varphi)$, either $E(\psi)$ belongs to this list, or both $\neg E(\psi)$ and $\neg \psi$ do.

We will see right away how we use some parts of this object to provide a basic optimisation.

3.4 Caching

The procedure for setting up initial boards can generate the same board twice. Consider the following two partitions from our previous example:

$$i, \varphi \mid j \qquad\qquad i \mid j, \varphi$$

Starting from both partitions, one can generate the following initial board: $\{\{i, \varphi\}, \{j, \varphi\}\}$.

In order to solve this problem, we use caching. For each input formula, $Cl(\varphi)$ is fixed, and so is the set of all possible Hintikka sets. So we can associate to each Hintikka set an integer. This is what we do in the *firstHSets* field of the *Board* object. Thus, each initial board is identified by the list of Hintikka set indexes, in increasing order. We store hashes of each initial board that has been already considered in order to avoid analysing the same game twice.

4 Translation-based Satisfiability Checking

The game-based approach to topological theorem proving embodied in HyLoBan uses Sustretov's reduction of the logics of T_0 and T_1 to relational semantics in the most direct way possible: by actually playing the PSPACE game he defines for the

relevant frame classes. But there is a simpler way of exploiting the reduction: with the help of the universal modality, we can encode the required frame conditions. Let's see how to do this.

Let's first consider $\mathsf{Log}(T_1)$. Let φ be a formula containing nominals $i_1, \ldots, i_k$. Then it is immediate that the formula

$$\psi_1(\varphi) = \varphi \wedge \bigwedge_{i=1}^{k} A(\Diamond i_k \rightarrow i_k)$$

is satisfiable on a finite relational **S4** model iff this model satisfies the condition T_1 from Definition 2.4, for all the nominals occurring in φ. After all, $A(\Diamond i_k \rightarrow i_k)$ is a direct statement of the T_1 condition: it clearly asserts (for every nominal occurring in φ) that all points named by nominals have no incoming arcs other than from themselves. In effect, we have used the universal modality to globally force the required constraint on models.

Matters are almost as straightforward for $\mathsf{Log}(T_0)$. Let $Nom(\varphi)$ be the set of nominals in φ. Then the formula:

$$\psi_0(\varphi) = \varphi \wedge \bigwedge_{i,j \in Nom(\varphi)} @_i \neg j \rightarrow (@_i \Box \neg j \vee @_j \Box \neg i)$$

is satisfiable on a finite relational **S4** model iff this model satisfies the condition T_0 from Definition 2.4, for all pairs of nominals occurring in φ. After all, the conjunction over these pairs systematically excludes non-trivial cycles involving the points named by these nominals. Once again we are using the universal modality to globally force the required constraint on models (recall that @ is defined using the universal modality).

Thus the following proposition holds:

Proposition 4.1

- *A formula φ belongs to $\mathsf{Log}(T_0)$ iff $\psi_0(\varphi) \rightarrow \varphi$ is valid on the class of **S4** frames.*
- *A formula φ belongs to $\mathsf{Log}(T_1)$ iff $\psi_1(\varphi) \rightarrow \varphi$ is valid on the class of **S4** frames.*

What does this give us? For a start, there is now a simpler proof of the PSPACE completeness of the logics of T_0 and T_1. After all, the logic of **S4** frames in hybrid logic enriched with the universal modality is known to be PSPACE complete (see [3]), and we have just encoded T_0 and T_1 validity in this logic.

More to the point for present purposes, however, is the fact that it gives us a new approach to hybrid topological theorem proving. Given a hybrid logic prover that can handle **S4** and the universal modality, the previous proposition gives us a simple recipe for using it for topological theorem proving purposes. Fortunately, such a prover exists, namely HyLoTab [18] [2] . Hence, armed with HyLoTab, we have a second way of doing topological theorem proving, one we can compare with HyLoBan.

---

[2] A referee asked if the description logic prover FaCT++ (see http://owl.man.ac.uk/factplusplus/) could be used instead. Indeed, it seems to provide everything we need: the $\mathcal{O}$ (one-of) operator could play the role of nominals, the TBox could play the role of the universal modality, and the prover does handle transitive roles. It is not clear to us if the present version handles reflexive roles. We are contacting the developers about this issue, and in the case of a possitive answer we will include FaCT++ in our future experiments.
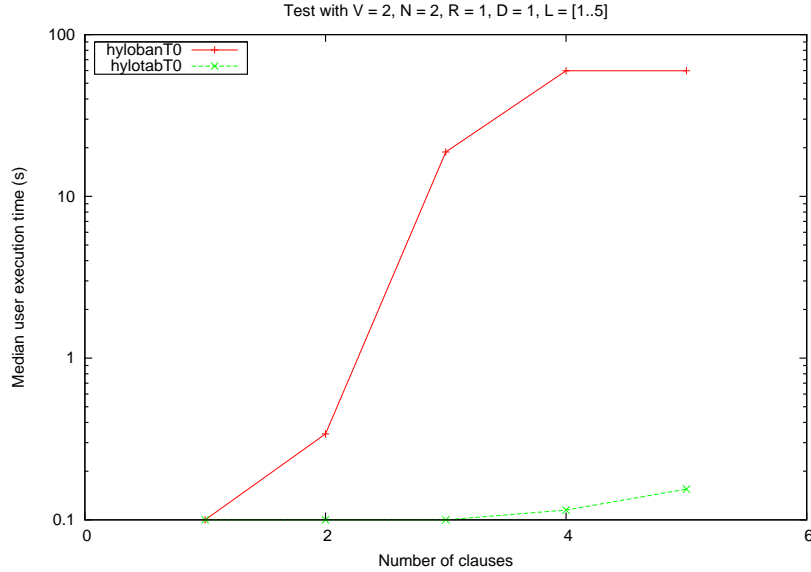
Figure 2. Median time versus size of formulas, for SAT test with T_0 axiom, between HyLoBan and HyLoTab

5 Performance Evaluation

We shall now evaluate the performance of the two approaches. After implementing the T_0 and T_1 translation-based satisfiability tests using HyLoTab[3], we compared it with HyLoBan's game-based approach; the chart in Figure 2 is for formulas with the T_0 axiom, and Figure 3 is for formulas with the T_1 axiom. These charts were obtained by running HyLoBan and HyLoTab on batches of random formulas of the language described in Section 2; the formulas contained 2 propositional symbols, 2 nominals, 1 relational symbol, and had a modal depth of 1. The formulas ranged from size 1 to size 5 in the number of conjunctions of clauses.

As we can clearly see, HyLoBan's performance is poor: even though it guarantees termination, HyLoBan median time is much higher than HyLoTab's. On the other hand, the tests also showed that there are simple formulas on which HyLoTab timed out, but which HyLoBan was able to solve. For example, the formula

$$\neg p \wedge A(p \vee \Diamond(\neg p \wedge n))$$

makes HyLoTab loop, while HyLoBan instantly claims its satisfiability with respect to the T_0 axiom.

We have identified one main performance weakness in HyLoBan, namely the way we generate Hintikka sets. Currently we generate *all* Hintikka sets that contain a formula ψ. We could instead only generate all sets that contain $\psi \cup csq(\psi)$, where $csq(\psi)$ is a set of "consequence" formulas obtained by running a simplified tableaux algorithm on the formula ψ. These consequence formulas might be, for example, the set of formulas present in a branch of the tableaux algorithm without having used a branching rule. Using such a combination of the game and tableaux-based approached we may be able to get both better performance and guarantee

---

[3] This modified version of HyLoTab is available at http://trac.loria.fr/projects/hylotab
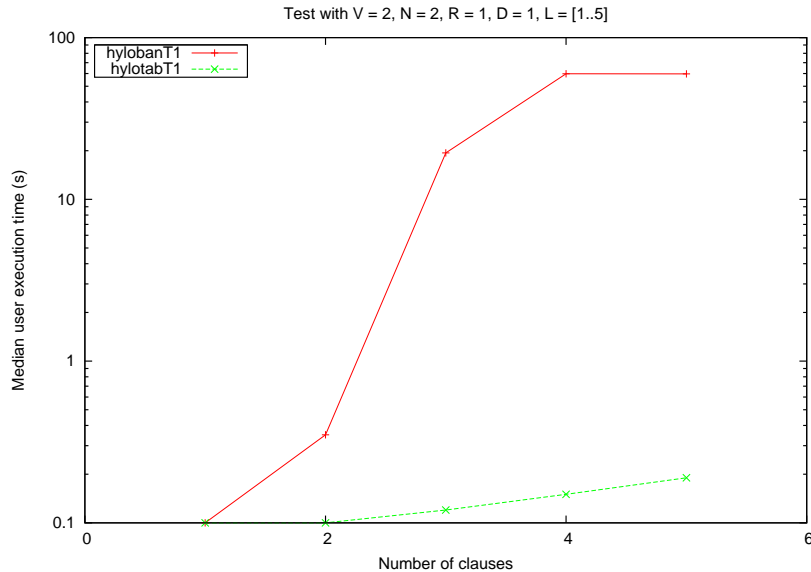
Figure 3. Median time versus size of formulas, for SAT test with T_1 axiom, between HyLoBan and HyLoTab

termination.

Another optimisation would be to use an auxiliary tableaux algorithm to remove parts of the input formula that are already unsatisfiable in weaker hybrid logics. For example, if the input formula is $\varphi \vee \psi$, and if we can prove that ψ is unsatisfiable in a weaker hybrid logic for which a terminating prover exists (in particular, the minimal hybrid logic $\mathbf{K}$), then we can simply launch the game-based prover on φ.

6   Conclusion

In this paper we have discussed two preliminary experiments in theorem proving for topological hybrid logic; the long term goal of these experiments is to integrate such theorem proving into the InToHyLo [2] framework.

As the evaluation clearly shows, the current version of the game-based approach implemented in HyLoBan is inferior to the translation-based approach using the universal modality. But we believe that it is worth experimenting further with the game-based approach. For a start, there are a number of obvious optimisations which could be built into the system. Furthermore, HyLoBan is essentially a *generic* game-based theorem proving tool. In our view, such a tool could be a useful addition to the InToHyLo framework. For example, we believe it may be useful for experimenting with theorem proving for hybrid *neighbourhood* logics (see [7] for some preliminary work on such logics).

Be that as it may, the current best-bet for better topological hybrid theorem prover lies with the translation-based approach. And it seems clear that the performance of this approach can be much enhanced. For a start, as we have already noted, the description logic prover FaCT++ might offer us everything we need; if it can handle reflexive roles then it will surely be a strong candidate for an efficient prover for topological hybrid logics. Moreover, the first version of HTab, a terminating tableau prover for hybrid logic was recently implemented (see [9,10]).

11

This new prover convincingly outperforms HyLoTab for the basic logic **K**, and we believe it will be straightforward to incorporate into HyLoTab recently announced terminating tableaux algorithms which covers hybrid **S4** enriched with the universal modality (see [5]). This seems likely to lead to substantial performance gains, and hope to run HTab-based experiments on topological theorem proving shortly.

References

[1] M. Aiello, J. van Benthem, and G. Bezhanishvili. Reasoning about space: The modal way. *Journal of Logic and Computation*, 13(6):889–929, 2003.

[2] C. Areces, P. Blackburn, D. Gorín, and G. Hoffmann. Inference tools for hybrid logics (InToHyLo). Manuscript, LORIA, available from http://www.loria.fr/~areces, 2007.

[3] C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8(5):653–679, 2000.

[4] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logics*. Elsevier, 2006.

[5] T. Bolander and P. Blackburn. Terminating tableaux calculi for hybrid logics extending **K**. Submitted to *Method For Modalities*, 2007.

[6] L. Esakia. Diagonal constructions, Löb's formula, and Cantor's scattered spaces. In *Studies in Logic and Semantics*, pages 128–143. Metsniereba, 1981. In Russian.

[7] D. Figueira. Bisimulation and complexity for neighbourhood semantics. Technical report, LORIA, 2007.

[8] GHC, The Glasgow Haskell Compiler. http://www.haskell.org/ghc/. Last visited: 15/09/07.

[9] G. Hoffmann. HTab: Terminating tableaux system for hybrid logic. Master's thesis, Département de formation doctorale en informatique, UFR STMIA. École doctorale IAEM Lorraine, 2007. (English Version).

[10] G. Hoffmann and C. Areces. HTab: a terminating tableaux system for hybrid logic. Submitted to *Method For Modalities*, 2007.

[11] S. Peyton Jones and J. Hughes (editors). Haskell 98: A non-strict, purely functional language. Technical report, Haskell.org, 1999.

[12] J. McKinsey and A. Tarski. The algebra of topology. *Annals of Mathematics*, 45:141–191, 1944.

[13] M. Orlov. Efficient generation of set partitions. Technical report, Faculty of Engineering and Computer Sciences, University of Ulm, 2002. http://www.cs.bgu.ac.il/~orlovm/papers/partitions.pdf.

[14] V. Shehtman. Modal logics of domains of the real plane. *Studia Logica*, 42:63–80, 1983.

[15] D. Sustretov. Topological semantics and decidability. http://www.arxiv.org/abs/math/0703106.

[16] D. Sustretov. Topological semantics and decidability. In J. Villadsen, T. Bolander, and T. Braüner, editors, *International Workshop on Hybrid Logic 2007 (HyLo 2007)*, Dublin, Ireland, 2007.

[17] A. Tarski. Der Aussagenkalkül und die Topologie. *Fund. Math.*, 31:103–134, 1938.

[18] J. van Eijck. HyLoTab — Tableau-based theorem proving for hybrid logics. Manuscript, CWI, available from http://www.cwi.nl/~jve/hylotab, 2002.