A Straightforward Saturation-Based Decision Procedure for Hybrid Logic

Mark Kaminski[1] Gert Smolka[1]

Programming Systems Lab
Saarland University
Saarbrücken, Germany

Abstract

In this paper we present a saturation-based decision procedure for basic hybrid logic extended with the universal modality. Termination of the procedure is guaranteed by constraints that are conceptually simpler than the loop-checks commonly used with related tableau-based decision methods in that they do not rely on the order in which new formulas are introduced. At the same time, our constraints allow us to limit the worst-case asymptotic complexity of the procedure more tightly than it seems to be possible for methods using conventional loop-checks. The procedure is based on Hardt and Smolka's higher-order formulation of hybrid logic [10].

1 Introduction

Recently, several tableau-based decision procedures, both prefixed and internalized [7, 6], were developed for $\mathcal{H}(E)$, the basic hybrid language [5] extended with the universal modality. Termination of the procedures is guaranteed by restricting the applicability of rules that generate new prefixes (or nominals, respectively) to formulas prefixed by so-called *urfathers*. A prefix on a tableau branch is called an urfather of that branch if the associated set of formulas is not included in the corresponding set for any other prefix that occurs earlier on the branch.

We present a saturation-based decision procedure for $\mathcal{H}(E)$ based on the higher-order formulation of hybrid logic devised by Hardt and Smolka [10]. The procedure is obtained from a model existence theorem as a set of terminating saturation rules that, applied to a set C of hybrid formulas, will construct a model if and only if C is satisfiable. Like in internalized tableau systems [4, 7, 6], the saturation rules do not rely on any extensions of the object syntax. Termination of the procedure is guaranteed by constraints restricting the expansion of diamond-prefixed formulas, reminiscent of the urfather-based loop-checks in [7, 6], but motivated by a different model construction. Our algorithm works on unordered sets of formulas rather than ordered tableau branches and treats equality explicitly. Given a set of formulas, the admissibility of diamond expansion does not depend on any knowledge about how the set was constructed. Moreover, since our expansion constraints only depend on very specific subsets of the formulas that are true at a

[1]{kaminski,smolka}@ps.uni-sb.de

17.5.2007

$$\begin{array}{rcl@{\qquad}rcl}
\mathring{f} & := & f\pi & @_u t & := & (\lambda\pi.t)u \\
\mathring{u} & := & \pi\doteq u & \mathring{E}t & := & E(\lambda\pi.t) \\
\mathring{\Diamond}t & := & \Diamond\pi(\lambda\pi.t) & \mathring{A}t & := & A(\lambda\pi.t) \\
\mathring{\Box}t & := & \Box\pi(\lambda\pi.t) & & &
\end{array}$$

Figure 1: Hybrid Notation in Higher-Order Syntax

given nominal rather than on all such formulas, we are able to give upper bounds for the asymptotic worst-case complexity of our procedure that are smaller than the ones known for conventional loop-checks. We expect our approach to provide for a simple and comparatively efficient implementation of the decision procedure.

2 Basics

As the formal basis for our presentation we take the formulation of hybrid logic based on the simply typed λ-calculus introduced in [10]. For an introduction to the simply typed λ-calculus, see [3].

For each type T we assume a countably infinite set Var_T of variables, and define $Var := \bigcup_T Var_T$. Two base types are given special interpretation: the type V of vertices and the type B of truth values. Variables of type V are called *nominal variables*, and are written as π, x, y, z. Variables $f, g : \mathsf{V} \to \mathsf{B}$ are called *propositional variables*. We also assume a countably infinite set Par of constants of type V, which we call *parameters*. Parameters are written as a, b. Elements of the set $Nam := Par \cup Var$ are called *names*. Names of type V are written as u, v, w.

We consider λ-terms, written as s, t, over the following signature:

- Type constants B, V
- Boolean connectives $\vee, \wedge : \mathsf{B} \to \mathsf{B} \to \mathsf{B}$ and $\neg : \mathsf{B} \to \mathsf{B}$
- Equality predicate on vertices $\doteq : \mathsf{V} \to \mathsf{V} \to \mathsf{B}$
- Relational constant $R : \mathsf{V} \to \mathsf{V} \to \mathsf{B}$
- Modal operators $\Diamond, \Box : \mathsf{V} \to (\mathsf{V} \to \mathsf{B}) \to \mathsf{B}$
- Universal modalities $E, A : (\mathsf{V} \to \mathsf{B}) \to \mathsf{B}$

For a term t, $\mathcal{V}t$ denotes the set of nominal variables that occur free in t, and $\mathcal{N}t := \mathcal{V}t \cup \{a \in Par \mid a \text{ occurs in } t\}$. If $\mathcal{V}t = \varnothing$, t is called *closed*.

Terms of type B are called *formulas*. We denote the set of all formulas by For. Given a term t, we write $\mathcal{F}t$ for the set of all subformulas of t. We call a formula t *monadic* if for every subterm s it holds $|\mathcal{V}s| \leq 1$, and every subterm of the form $\lambda x.t$ is closed. It can be shown that monadic formulas of the form
$t \in \mathcal{MFI} ::= fu \mid u\doteq v \mid \neg t \mid t \vee t \mid \Diamond u(\lambda\pi.t)$
naturally correspond to formulas of $\mathcal{H}(@)$ [10]. It is easily seen that the monadic fragment of $\mathcal{MFI}$ extended by terms of the form $E(\lambda\pi.t)$ analogously corresponds to $\mathcal{H}(E)$. Let us call formulas which are elements of this extended set *proper*. Figure 1 summarizes the correspondence between hybrid and higher-order syntax. Note that a proper formula never contains subterms of the form Ruv. A set C of β-normal, η-long, negation-normal formulas is called a *clause* if every element $t \in C$ is either proper or of the form Ruv. In other words, a clause contains only

formulas of the form Ruv and proper formulas of the form
$p ::= fu \mid \neg fu \mid u \doteq v \mid \neg u \doteq v \mid p * p \mid \mu u(\lambda\pi.p) \mid M(\lambda\pi.p)$
where $* \in \{\wedge, \vee\}, \mu \in \{\Diamond, \Box\}, M \in \{A, E\}$. A clause C is called proper if every formula $t \in C$ is proper. C is called monadic if every proper $t \in C$ is monadic. $\mathcal{V}$, $\mathcal{N}$, and $\mathcal{F}$ are extended to clauses in the natural way. So, for instance, $\mathcal{N}C := \bigcup_{t \in C} \mathcal{N}t$. Please note that in the following, we will always think of clauses as conjunctions, not as disjunctions.

Definition (Interpretation) An *interpretation* of hybrid logic is a standard interpretation of the simply typed λ-calculus that interprets the type constants B, V, the parameters $\doteq, \neg, \wedge, \vee, R, \Diamond, \Box, E, A$, and the variables in *Var* such that $\mathcal{D}\mathsf{B} = \{0, 1\}$ where $0 \neq 1$, $\mathcal{D}\mathsf{V} \neq \varnothing$, and

$$\begin{aligned}
\mathcal{D}(u \doteq v) = 1 &\iff \mathcal{D}u = \mathcal{D}v \\
\mathcal{D}(\neg t) = 1 &\iff \mathcal{D}t \neq 1 \\
\mathcal{D}(s \wedge t) = 1 &\iff \mathcal{D}s = 1 \text{ and } \mathcal{D}t = 1 \\
\mathcal{D}(s \vee t) = 1 &\iff \mathcal{D}s = 1 \text{ or } \mathcal{D}t = 1 \\
\mathcal{D}(Ruv) = 1 &\iff (\mathcal{D}u, \mathcal{D}v) \in \mathcal{D}R \\
\mathcal{D}(\Diamond ut) = 1 &\iff \exists\, a \in \mathcal{D}\mathsf{V} : (\mathcal{D}u, a) \in \mathcal{D}R \text{ and } \mathcal{D}ta = 1 \\
\mathcal{D}(\Box ut) = 1 &\iff \forall\, a \in \mathcal{D}\mathsf{V} : (\mathcal{D}u, a) \in \mathcal{D}R \text{ implies } \mathcal{D}ta = 1 \\
\mathcal{D}(Et) = 1 &\iff \exists\, a \in \mathcal{D}\mathsf{V} : \mathcal{D}ta = 1 \\
\mathcal{D}(At) = 1 &\iff \forall\, a \in \mathcal{D}\mathsf{V} : \mathcal{D}ta = 1
\end{aligned}$$

Whenever we have $\mathcal{D}t = 1$, we also write $\mathcal{D} \vDash t$ and say that $\mathcal{D}$ *satisfies* t, or that t is *valid* in $\mathcal{D}$. A term is called *satisfiable* if it has a satisfying interpretation.

We write $(\lambda x.t){\downarrow}u$ for $t[x := u]$ and make use of the following property.

Proposition 2.1 (β-Compatibility) $\mathcal{D}(s{\downarrow}t) = \mathcal{D}s(\mathcal{D}t)$.

3 Saturatedness and Model Existence

For every clause C, let $\sim_C$ denote the equivalence closure of the relation $\{(u, v) \mid u \doteq v \in C\}$. We also write $[u]_C$ to denote the set $\{v \mid u \sim_C v\}$.

Let $\iota \in \mathcal{P}(\mathit{Nam}) \to \mathit{Nam}$ be a choice function, i.e. a function such that, for all $A \in \mathcal{P}(\mathit{Nam})$: $|A| > 0 \implies \iota A \in A$.

For every clause C we define $\rho_C \in \mathcal{N}C \to \mathcal{N}C$ such that $\rho_C u = \iota[u]_C$. Names u such that $\rho_C u = u$ are called *ρ_C-normal.*

So, given a name u, ρ_C returns a canonical representative of the equivalence class of u in C. Later, when we look at the saturation process, it will be obvious that ρ_C needn't be re-computed from scratch for every intermediate clause, but can be constructed incrementally, for instance using a disjoint-set forest (see [9, 8]).

Definition (Triviality) A clause C is called *trivial* if either

- $\neg u \doteq v \in C$ for some u, v such that $\rho_C u = \rho_C v$, or
- $fu \in C$ and $\neg fv \in C$ for some u, v, f such that $\rho_C u = \rho_C v$.

Formulas that are either of the form $\Diamond ut$ or $\Box ut$ are called *modal literals.*

Let C be given. We define the notation

$$P_C u := \{\lambda x.\Diamond xt \mid \Diamond ut \in C\} \cup \{\lambda x.\Box xt \mid \Box ut \in C\}$$
$$P_C(\Diamond ut) := \begin{cases} \{\lambda x.\Diamond xt\} \cup \{\lambda x.\Box xs \mid \Box us \in C\} & \text{if } \Diamond ut \in C \\ \varnothing & \text{otherwise} \end{cases}$$

($\mathcal{S}_{\dot=}$) $P_C u \subseteq P_C(\rho_C u)$.
($\mathcal{S}_\wedge$) If $s \wedge t \in C$, then $s, t \in C$.
($\mathcal{S}_\vee$) If $s \vee t \in C$, then $s \in C$ or $t \in C$.
($\mathcal{S}_\Diamond$) If $\Diamond ut \in C$ and $\rho_C u = u$,
then $\exists v : P_C(\Diamond ut) \subseteq P_C(\Diamond vt)$ and $\Diamond vt$ is expanded in C.
($\mathcal{S}_\Box$) If $\Box ut, Ruv \in C$, then $t{\downarrow}v \in C$.
($\mathcal{S}_A$) If $At \in C$ and $\rho_C u = u$, then $t{\downarrow}u \in C$.
($\mathcal{S}_E$) If $Et \in C$, then $t{\downarrow}u \in C$ for some u.

Figure 2: Saturatedness Conditions

The sets $P_C u$ and $P_C(\Diamond ut)$ are called the *patterns* of u and $\Diamond ut$, respectively, the latter also being called *diamond patterns*. A formula of the form $\Diamond ut \in C$ is called *expanded* in C if there exists a name v such that $Ruv \in C$ and $t{\downarrow}v \in C$.

We use patterns to abstract away names from formulas in such a way that every two modal literals of the form μut and μvt, for some $\mu \in \{\Diamond, \Box\}$, correspond to the same abstraction $\lambda x.\mu xt$. Although diamond expansion introduces new nominal variables and hence formulas that are not subformulas of the terms from which they were generated, such formulas can always be related to a subterm of a generating formula by abstracting away the newly introduced variable. As we will see in Chapter 7, this property is of crucial importance when it comes to showing termination of our procedure.

Definition (Saturatedness) A clause C is called *saturated* if it satisfies $\mathcal{S}_{\dot=}$, $\mathcal{S}_\wedge, \mathcal{S}_\vee, \mathcal{S}_\Box, \mathcal{S}_A, \mathcal{S}_E$, and $\mathcal{S}_\Diamond$.

The saturatedness conditions given in Figure 2 are mostly straightforward, so let us just explain the intuition behind $\mathcal{S}_\Diamond$. There we observe that, provided there is some name w such that $Rvw, t{\downarrow}w \in C$ and $P_C(\Diamond ut) \subseteq P_C(\Diamond vt)$, there is no need to expand $\Diamond ut$ because in a model of C the vertex corresponding to the introduced name could not make any formulas true that are not already true in the vertex corresponding to w. Therefore, given a model of $P_C(\Diamond vt)$, it suffices to introduce an edge between the vertices corresponding to u and v to ensure that the resulting model satisfies $P_C(\Diamond ut)$. Note also that $\mathcal{S}_\Diamond$ and $\mathcal{S}_A$ apply only to ρ_C-normal names. We take these names to be the vertices of our model. The precise model construction is as follows.

Given a saturated clause C, let $\mathcal{D}_C$ be an interpretation such that

$$\begin{aligned}
\mathcal{D}_C\mathsf{V} &= \{\rho_C u \mid u \in \mathcal{N}C\} \\
\mathcal{D}_C u &= \rho_C u \\
\mathcal{D}_C f &= \lambda u \in \mathcal{D}_C\mathsf{V}.\, \exists v : \rho_C v = u \wedge fv \in C \\
\mathcal{D}_C R &= \{(u, \rho_C v) \mid \rho_c u = u \wedge \\
&\qquad\qquad \exists \Diamond ut \in C, w : P_C(\Diamond ut) \subseteq P_C(\Diamond wt) \wedge Rwv \in C\}
\end{aligned}$$

Looking at $\mathcal{D}_C$, one may wonder why we do not weaken $\mathcal{S}_\Box$ to

($\mathcal{S}'_\Box$) If $\Box ut, Ruv \in C$ and $\rho_C u = u$, then $t{\downarrow}v \in C$.

Condition $\mathcal{S}'_\Box$ looks more similar to $\mathcal{S}_\Diamond$ and $\mathcal{S}_A$, and seems sufficient because names that are not ρ_C-normal do not directly appear in the model. However, this intuition turns out to be wrong.

Consider the clause $C := \{\Diamond u(\lambda\pi.f\pi), Ruw, fw, \Box u(\lambda\pi.\neg f\pi), \Diamond v(\lambda\pi.f\pi), \Box v(\lambda\pi.\neg f\pi)\}$ and assume $\rho_C u = v$. If we replace $\mathcal{S}_\Box$ with $\mathcal{S}'_\Box$, C becomes saturated. But C is clearly unsatisfiable. We see that since $\mathcal{S}_\Diamond$ does not require $\Diamond v(\lambda\pi.f\pi)$ to be expanded, we have to enforce propagation of $\Box u(\lambda\pi.\neg f\pi)$ along Ruw to arrive at a trivial clause.

Theorem 1 (Model Existence) *If C is a non-trivial saturated clause and $t \in C$ proper, then $\mathcal{D}_C \vDash t$.*

Proof By induction on $|t|$.

Case $t = fu$. Assume $fu \in C$. Then $\mathcal{D}_C(fu) = \mathcal{D}_C f(\rho_C u) = 1$ by the definition of $\mathcal{D}_C f$.

Case $t = \neg fu$. Assume $\neg fu \in C$. Since C non-trivial, there exists no v such that $\rho_C v = \rho_C u$ and $f(\rho_C v) \in C$, i.e. $\mathcal{D}_C f(\rho_C u) = \mathcal{D}_C(fu) \neq 1$. Hence $\mathcal{D}_C \vDash \neg fu$.

Case $t = u \dot{=} v$. Assume $u \dot{=} v \in C$. Then $\mathcal{D}_C u = \rho_C u = \rho_C v = \mathcal{D}_C v$, i.e. $\mathcal{D}_C \vDash u \dot{=} v$.

Case $t = \neg u \dot{=} v$. Assume $\neg u \dot{=} v \in C$. Since C is non-trivial, $\rho_C u \neq \rho_C v$, i.e. $\mathcal{D}_C \nvDash u \dot{=} v$. Hence $\mathcal{D}_C \vDash \neg u \dot{=} v$.

Case $t = t_1 \wedge t_2$. Assume $t_1 \wedge t_2 \in C$. By $\mathcal{S}_\wedge$, $t_1 \in C$ and $t_2 \in C$. By the inductive hypothesis, $\mathcal{D}_C \vDash t_1$ and $\mathcal{D}_C \vDash t_2$. Therefore $\mathcal{D}_C \vDash t_1 \wedge t_2$.

Case $t = t_1 \vee t_2$. Analogously to the preceding case.

Case $t = \Diamond us$. Assume $\Diamond us \in C$. Then by $\mathcal{S}_{\dot{=}}$ it holds $\Diamond(\rho_C u)s \in C$. Since $\rho_C(\rho_C u) = \rho_C u$, by $\mathcal{S}_\Diamond$ there exist v, w such that $P_C(\Diamond(\rho_C u)s) \subseteq P_C(\Diamond vs)$ and $Rvw, s{\downarrow}w \in C$. Then $(\mathcal{D}_C u, \mathcal{D}_C w) = (\rho_C u, \rho_C w) \in \mathcal{D}_C R$. Moreover, by the inductive hypothesis and β-compatibility, $\mathcal{D}_C s(\mathcal{D}_C w) = \mathcal{D}_C(s{\downarrow}w) = 1$. We have shown that $\mathcal{D}_C w$ witnesses validity of $\Diamond us$.

Case $t = \Box us$. Assume $\Box us \in C$. We have to show that for every pair $(v, w) \in \mathcal{D}_C R$ such that $v = \mathcal{D}_C u = \rho_C u$ it holds $\mathcal{D}_C sw = 1$.
So assume, for some $v \in \mathcal{D}_C \mathsf{V}$, that $(\rho_C u, v) \in \mathcal{D}_C R$. Then C contains a modal literal of the form $\Diamond(\rho_C u)s'$ such that, for some u' and w with $\rho_C w = v$ it holds $P_C(\Diamond(\rho_C u)s') \subseteq P_C(\Diamond u's')$ and $Ru'w \in C$. By $\mathcal{S}_{\dot{=}}$ it holds $\Box(\rho_C u)s \in C$. Consequently, $\Box u's \in C$. Now by $\mathcal{S}_\Box$, $s{\downarrow}w \in C$. By the inductive hypothesis and β-compatibility it holds $\mathcal{D}_C sv = \mathcal{D}_C s(\rho_C w) = \mathcal{D}_C s(\mathcal{D}_C w) = \mathcal{D}_C(s{\downarrow}w) = 1$.

Case $t = As$. Assume $As \in C$. To show: $\mathcal{D}_C su = 1$ for all $u \in \mathcal{D}_C \mathsf{V}$. So let $u \in \mathcal{D}_C \mathsf{V}$ be arbitrary. Since $\rho_C u = u$, by $\mathcal{S}_A$, $s{\downarrow}u \in C$. By the inductive hypothesis and β-compatibility, $\mathcal{D}_C(s{\downarrow}u) = \mathcal{D}_C s(\mathcal{D}_C u) = \mathcal{D}_C su = 1$.

Case $t = Es$. Assume $Es \in C$. By $\mathcal{S}_E$, $s{\downarrow}u \in C$ for some u. By the inductive hypothesis and β-compatibility, $\mathcal{D}_C(s{\downarrow}u) = \mathcal{D}_C s(\mathcal{D}_C u) = 1$, i.e. $\mathcal{D}_C u$ witnesses validity of Es. $\Box$

So, given a proper clause C, in order to show C satisfiable it suffices to find a non-trivial saturated clause $D \supseteq C$. Theorem 1 then guarantees that $\mathcal{D}_D \vDash C$.

4 Saturation Algorithm

Definition (Saturation Relation) The saturation relation $\to$ on clauses is defined such that $C \to D$ if and only if $C \subsetneq D$ and D can be obtained from C by one of the rules $\mathcal{C}_{\dot{=}}, \mathcal{C}_\wedge, \mathcal{C}_\vee, \mathcal{C}_\Box, \mathcal{C}_A$, $\mathcal{C}_E$, or $\mathcal{C}_\Diamond$.

($\mathcal{C}^{\Diamond}_{\doteq}$) If $\Diamond ut \in C$, add $\Diamond(\rho_C u)t$.
($\mathcal{C}^{\Box}_{\doteq}$) If $\Box ut \in C$, add $\Box(\rho_C u)t$.
($\mathcal{C}_\wedge$) If $s \wedge t \in C$, add s and t.
($\mathcal{C}_\vee$) If $s \vee t \in C$ and neither $s \in C$ nor $t \in C$, add s or t.
($\mathcal{C}_\Diamond$) If $\Diamond ut \in C$, $\rho_C u = u$, and
there is no v such that $P_C(\Diamond ut) \subseteq P_C(\Diamond vt)$ and $\Diamond vt$ is expanded in C,
add Rux and $t{\downarrow}x$ for some $x \notin \mathcal{V}C$.
($\mathcal{C}_\Box$) If $\Box ut, Ruv \in C$, add $t{\downarrow}v$.
($\mathcal{C}_A$) If $At \in C$ and $\rho_C u = u$, add $t{\downarrow}u$.
($\mathcal{C}_E$) If $Et \in C$ and for no $u \in \mathcal{N}C$, $t{\downarrow}u \in C$, add $t{\downarrow}x$ for some $x \notin \mathcal{V}C$.

Figure 3: Saturation Rules

A clause C is called *terminal* if there exists no D such that $C \to D$. We say $C \to D$ *don't care* if $C \to D$ and D is obtained from C by one of the rules excluding $\mathcal{C}_\vee$. We say $C \to D_1, D_2$ *don't know* if D_1, D_2 are the two alternative results of applying $\mathcal{C}_\vee$ to some formula in C.

Proposition 4.1 (Soundness) *1. If $C \to D$ don't care, then C is satisfiable if and only if D is satisfiable.*

2. If $C \to D_1, D_2$ don't know, then C is satisfiable if and only if D_1 is satisfiable or D_2 is satisfiable.

We can now describe the proposed decision procedure. Given a finite monadic and proper clause C, we first construct $C_0 := C \cup \{\pi \doteq \pi\}$. Using saturation and the usual backtracking techniques, we search for a saturated non-trivial clause D such that $C_0 \to^* D$. C is satisfiable if and only if D exists.

The reader might be wondering why we need to construct C_0 and cannot saturate C directly. The reason is that we need to ensure that at least one name of type V occurs free in the initial clause. This way we prevent the saturation relation from terminating with clauses like $\{A(\lambda\pi.g\pi \wedge \neg g\pi)\}$, that are only satisfiable in an empty model, i.e. an interpretation $\mathcal{D}$ such that $\mathcal{D}\mathsf{V} = \varnothing$.

5 Example

Let us demonstrate the basic properties of the saturation algorithm and the model construction with an example. Consider the following input clause C: $\{A(\lambda\pi.\Diamond\pi(\lambda\pi.f\pi)), \Box\pi(\lambda\pi.\Diamond\pi(\lambda\pi.\pi \doteq a))\}$. (In hybrid notation, C can be written more concisely as $\{\mathring{A}\mathring{\Diamond}\mathring{f}, \mathring{\Box}\mathring{\Diamond}\mathring{a}\}$.) From C we construct $C_0 := C \cup \{\pi \doteq \pi\}$ and proceed as follows:

$$\begin{array}{lcl}
C_0: & A(\lambda\pi.\Diamond\pi(\lambda\pi.f\pi)), \Box\pi(\lambda\pi.\Diamond\pi(\lambda\pi.\pi \doteq a)), \pi \doteq \pi & \\
C_2: & \Diamond\pi(\lambda\pi.f\pi), \Diamond a(\lambda\pi.f\pi) & \mathcal{C}_A \text{ on } A(\ldots), \pi, a \\
C_3: & R\pi x, fx & \mathcal{C}_\Diamond \text{ on } \Diamond\pi(\lambda\pi.f\pi) \\
C_4: & \Diamond x(\lambda\pi.f\pi) & \mathcal{C}_A \text{ on } A(\ldots), x \\
C_5: & \Diamond x(\lambda\pi.\pi \doteq a) & \mathcal{C}_\Box \text{ on } \Box\pi(\ldots), R\pi x \\
C_6: & Rxy, y \doteq a & \mathcal{C}_\Diamond \text{ on } \Diamond x(\lambda\pi.\pi \doteq a)
\end{array}$$

Now assume that $\rho_{C_6} y = \rho_{C_6} a = a$. Then $\mathcal{C}_A$ is not applicable to $A(\ldots), y$. Since additionally $P_{C_6}(\Diamond x(\lambda\pi.f\pi)) = P_{C_6}(\Diamond a(\lambda\pi.f\pi)) \subseteq P_{C_6}(\Diamond\pi(\lambda\pi.f\pi))$, C_6 is saturated. The model constructed from C_6 looks as follows:

$$\begin{tikzcd}
\pi \arrow[r, "f"'] & x \arrow[loop above] \arrow[r, bend left] & a \arrow[l, bend left]
\end{tikzcd}$$

6 Completeness

Theorem 2 (Completeness) *Every terminal clause is saturated.*

Proof Assume C is a terminal clause. We have to show that C satisfies all the saturatedness conditions.

$\mathcal{S}_{\doteq}$**:** By assumption, C is closed under application of $\mathcal{C}^{\Diamond}_{\doteq}$ and $\mathcal{C}^{\Box}_{\doteq}$, i.e. for every $\mu ut \in C$ where $\mu \in \{\Diamond, \Box\}$ we have $\mu(\rho_C u)t \in C$. Then for every $\lambda x.\mu xt \in P_C u$ it holds $\lambda x.\mu xt \in P_C(\rho_C u)$, i.e. $P_C u \subseteq P_C(\rho_C u)$.

$\mathcal{S}_\wedge, \mathcal{S}_\vee, \mathcal{S}_\Box, \mathcal{S}_A$ are easily shown using $\mathcal{C}_\wedge, \mathcal{C}_\vee, \mathcal{C}_\Box, \mathcal{C}_A$, respectively.

$\mathcal{S}_E$**:** Assume $Et \in C$. Since every application of $\mathcal{C}_E$ enlarges $\mathcal{N}C$ by a new variable, terminality of C implies that $\mathcal{C}_E$ is not applicable. Then there has to exist some u such that $t{\downarrow}u \in C$.

$\mathcal{S}_\Diamond$**:** Assume $\Diamond ut \in C$ and $\rho_C u = u$. As with $\mathcal{C}_E$, terminality of C implies that $\mathcal{C}_\Diamond$ is not applicable. Then there has to exist some v such that $P_C(\Diamond ut) \subseteq P_C(\Diamond vt)$ and $\Diamond vt$ is expanded in C. $\Box$

7 Termination

Since $\mathcal{C}_\vee$ is the only rule that leads to "don't know"-reductions, the search tree our decision procedure has to traverse is finitely branching (binary, to be more precise). Therefore, to prove termination of the procedure it suffices to show that the tree has finite depth. This is clearly the case if the relation $\rightarrow$ always terminates. So, let us prove $\rightarrow$ terminating.

The *degree* of a clause C is defined as follows: $\deg C := \max_{t \in C} |t|$.

Lemma 7.1 *Let $C_0 \rightarrow C_1 \rightarrow \ldots$ be a saturation sequence. There exist no two indices $i, j \in \mathbb{N}$ such that $i < j$ and, for some $Et \in C_i \subseteq C_j$, both C_{i+1} and C_{j+1} are obtained by an application of $\mathcal{C}_E$ to Et.*

Proof Assume by contradiction i, j do exist. Since C_{i+1} was obtained by applying $\mathcal{C}_E$ to $\Diamond ut$, for some w we must have $t{\downarrow}w \in C_{i+1} \subseteq C_j$, contradicting the applicability of $\mathcal{C}_E$ in C_j. $\Box$

Proposition 7.2 *If $C \rightarrow D$, then $P_C(\Diamond ut) \subseteq P_D(\Diamond ut)$.*

Lemma 7.3 *Let $C_0 \rightarrow C_1 \rightarrow \ldots$ be a saturation sequence. There exist no two indices $i, j \in \mathbb{N}$, such that $i < j$ and*

1. *C_{i+1} is obtained by an application of $\mathcal{C}_\Diamond$ to $\Diamond ut \in C_i$,*
2. *C_{j+1} is obtained by an application of $\mathcal{C}_\Diamond$ to $\Diamond vt \in C_j$,*
3. *$P_{C_i}(\Diamond ut) = P_{C_j}(\Diamond vt)$.*

Proof Assume by contradiction indices i, j do exist. Clearly, $u \neq v$. By assumption, $\mathcal{C}_\Diamond$ is applicable to $\Diamond vt$ in C_j. This implies, amongst other things, that there exists no w such that $P_{C_j}(\Diamond vt) \subseteq P_{C_j}(\Diamond wt)$ and $\Diamond wt$ is expanded in C_j. But, also by assumption, $\Diamond ut$ is expanded in C_j and, by Prop. 7.2, $P_{C_j}(\Diamond ut) \supseteq P_{C_i}(\Diamond ut) = P_{C_j}(\Diamond vt)$. Contradiction. $\square$

Proposition 7.4 *If C is monadic and $C \to D$, then D is monadic.*

The following proposition is an analogue to the "quasi-subformula property" as stated in [7] or in [6].

Lemma 7.5 (Pattern Preservation) *Let C be monadic and $C \to D$.*

1. $At \in \mathcal{F}D \iff At \in \mathcal{F}C$
2. $Et \in \mathcal{F}D \iff Et \in \mathcal{F}C$
3. $\{\lambda x.\Diamond xt \mid \Diamond ut \in \mathcal{F}D\} = \{\lambda x.\Diamond xt \mid \Diamond ut \in \mathcal{F}C\}$
4. $\{\lambda x.\Box xt \mid \Box ut \in \mathcal{F}D\} = \{\lambda x.\Box xt \mid \Box ut \in \mathcal{F}C\}$

Proof By straightforward case analysis on the saturation rules. For the latter two claims we additionally observe that every two formulas of the form μut and μvt, $\mu \in \{\Diamond, \Box\}$, are abstracted to the same term $\lambda x.\mu xt$. $\square$

Proposition 7.6 *If $C \to D$, then* $\deg D = \deg C$.

Proposition 7.7 *There exists a function $f \in \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, exponential in the arguments, such that for every clause C it holds: If $\deg C, |\mathcal{N}C| < \infty$, then $|C| \leq f(\deg C, |\mathcal{N}C|)$.*

Theorem 3 (Termination) *$\to$ terminates on finite monadic clauses.*

Proof Let C_0 be a finite monadic clause. Assume by contradiction that there exists an infinite sequence $C_0 \to C_1 \to C_2 \to \ldots$. Since C_0 is finite, the sets $A_\Diamond := \{\{t\} \cup A \mid (t, A) \in \{\lambda x.\Diamond xs \mid \Diamond us \in \mathcal{F}C_0\} \times \mathcal{P}(\{\lambda x.\Box xs \mid \Box us \in \mathcal{F}C_0\})\}$ and $A_E := \{Et \in C_0\}$ are finite as well. By Prop. 7.4 and Lemma 7.5, for all C_i it holds $\{Et \in C_i\} \subseteq A_E$ and $\{P_{C_i}(\Diamond ut) \mid \Diamond ut \in C_i\} \subseteq A_\Diamond$. Now consider $C := \bigcup_{i \in \mathbb{N}} C_i$. By Lemma 7.1 and 7.3, C was obtained by finitely many applications of the rules $\mathcal{C}_E$ and $\mathcal{C}_\Diamond$, and hence $\mathcal{N}C$ is finite. Moreover, by Prop. 7.6, $\deg C = \deg C_0$. Therefore, by Prop. 7.7, C is finite. But since $C_i \to C_{i+1}$ implies $C_i \subsetneq C_{i+1}$, C must be infinite. Contradiction. $\square$

Since $\to$ terminates, it provides a basis for a decision procedure for the satisfiability problem in $\mathcal{H}(E)$. The procedure is in EXPTIME, which is known to be optimal for the problem [12, 1].

8 Discussion

Although our presentation is limited to a decision procedure for the class of all frames, the procedure can easily be modified to deal with other frame classes. Indeed, to obtain a decision procedure for most of the common frame classes it suffices to modify the rule $\mathcal{C}_\Box$, analogously to how it is demonstrated in [11]. Termination of the procedure is guaranteed as long as the modified rules do not violate Lemma 7.5. In particular, our expansion constraint suffices to deal with transitive frames.

Unlike urfather-based loop checks that are applicable on a per-prefix (or per-nominal) basis, the expansion constraint in $\mathcal{C}_\Diamond$ works on a per-diamond-pattern basis. It is possible for a clause to have two formulas of the form $\Diamond us, \Diamond ut$ such that one of them can be expanded and the other one cannot.

An interesting feature of our expansion constraint is that we only need to look at modal literals. This does not suffice in the case of urfather-based loop-checks, and can lead to more diamond expansions. Consider, for instance, a tableau representation of $\{\Diamond a(\lambda\pi.f\pi), ga, \Diamond b(\lambda\pi.f\pi), \neg gb\}$ (in hybrid notation, $\{@_a\mathring{\Diamond}\mathring{f}, @_a\mathring{g}, @_b\mathring{\Diamond}\mathring{f}, @_b\neg\mathring{g}\}$). Clearly, neither a nor b can be considered an urfather of the respective other name. Hence both diamonds need to be expanded. In our approach, saturatedness is achieved after only one expansion.

Even if one could restrict urfather-based loop-checks to consider modal literals only, our expansion constraint can still do better. Given m and n distinct formulas of the form $\Diamond ut$ and $\Box ut$, respectively, and assuming none of their proper subformulas are modal literals, there exist 2^{m+n} distinct sets of formulas, corresponding to up to approximately 2^{m+n} distinct states one can obtain using urfather-based loop-checks. In our case, however, the number of diamond expansions is bounded from above by $m \cdot 2^n$, the number of distinct diamond patterns. As an example, consider the formula $A(\lambda\pi. \bigvee_{J\subseteq I, |J|\geq 1} \bigwedge_{j\in J} \Diamond\pi(\lambda\pi.f_j\pi))$ (in hybrid notation, $\mathring{A}(\bigvee_{J\subseteq I,|J|\geq 1} \bigwedge_{j\in J} \mathring{\Diamond}\mathring{f}_j))$ where I is some index set and $f_1, \ldots, f_{|I|}$ are pairwise distinct. Depending on the strategy, urfather-based loop checking can lead to a number of diamond expansions that is exponential in $|I|$. Our expansion constraint, on the other hand, allows at most $|I|$ expansions.

9 Strong Diamond Expansion

Consider the following simplification of $\mathcal{C}_\Diamond$:

($\mathcal{C}^s_\Diamond$) If $\Diamond ut \in C$, $\rho_C u = u$, and $\Diamond ut$ is not expanded in C, add Rux and $t{\downarrow}x$ for some $x \notin \mathcal{V}C$.

We can define a saturation relation $\rightarrow_s$ analogously to $\rightarrow$, but with the rule $\mathcal{C}^s_\Diamond$ instead of $\mathcal{C}_\Diamond$.

Note that whenever $\mathcal{C}_\Diamond$ is applicable to a clause C, $\mathcal{C}^s_\Diamond$ is also applicable to C. The reverse direction does not hold, as we can see by looking at a simple example:

Let $C := \{\Diamond a(\lambda\pi.f\pi), \Diamond b(\lambda\pi.f\pi)\}$. Since C does not contain any equations, both a and b are ρ_C-normal. Moreover, $P_C(\Diamond b(\lambda\pi.f\pi)) = \{\lambda x.\Diamond x(\lambda\pi.f\pi)\} = P_C(\Diamond a(\lambda\pi.f\pi))$. So, $\mathcal{C}_\Diamond$ is not applicable to any of the formulas in C, whereas $\mathcal{C}^s_\Diamond$ is applicable to $\Diamond b(\lambda\pi.f\pi)$.

On formulas of the basic hybrid language $\mathcal{H}(@)$ even the simpler saturation relation $\rightarrow_s$ can be proven terminating by using essentially the same chain of reasoning as found in [6] for a related internalized calculus. Since applicability of $\mathcal{C}_\Diamond$ implies that of $\mathcal{C}^s_\Diamond$, our model construction from Section 3 suffices to show the completeness of the procedure based on $\rightarrow_s$.

10 Conclusion

Compared to the known tableau-based methods in [7, 6], our decision procedure has several noteworthy differences:

1. The applicability of saturation rules in our decision procedure does not depend on the order in which new formulas are added to a clause. This information is always present in the structure of a tableau, and is exploited by the tableau algorithms to ensure termination, usually by means of additional rule applicability conditions. In the saturation-based setting the same information could be represented by a saturation sequence $C_0 \to C_1 \to \ldots \to C$. Our saturation rules depend solely on the contents of a clause, not on how it was constructed.
2. The rule $\mathcal{C}_A$ is only applicable to ρ_C-normal names. Assuming ρ is implemented such that certain monotonicity principles are preserved, it will usually be possible to arrive at a saturated clause without having to propagate universal modalities to every name. A suitable, efficient implementation of ρ may be found in [8].
3. In contrast to loop-checks as used in [7, 6], which restrict expansion on a per-prefix basis, $\mathcal{C}_\diamond$ uses termination checking on a per-diamond-pattern basis. As demonstrated in Section 8, an input clause always generates strictly, and up to exponentially, fewer distinct diamond patterns than sets of subformulas. Thus we hope that in practice our procedure will perform significantly better than a procedure based on loop-checks.

We see the procedure as a promising basis for automated theorem proving in hybrid logic and hope for it to provide a practical alternative to other approaches (in particular to the direct resolution method by Areces, de Nivelle and de Rijke [2]).

References

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

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

[3] Barendregt, H. P. Lambda calculi with types. In *Handbook of Logic in Computer Science*, S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, Eds., vol. 2. Oxford University Press, 1992.

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

[5] Blackburn, P., and Seligman, J. Hybrid languages. *Journal of Logic, Language and Information 4*, 3 (1995), 251–272.

[6] Bolander, T., and Blackburn, P. Termination for hybrid tableaus. To appear in *Journal of Logic and Computation*.

[7] Bolander, T., and Braüner, T. Tableau-based decision procedures for hybrid logic. *Journal of Logic and Computation 16*, 6 (2006), 737–763.

[8] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[9] Galler, B. A., and Fisher, M. J. An improved equivalence algorithm. *Communications of the ACM 7*, 5 (May 1964), 301–303.

[10] Hardt, M., and Smolka, G. Higher-order syntax and saturation algorithms for hybrid logic. In *International Workshop on Hybrid Logic 2006 (HyLo 2006)*, to appear in ENTCS, Elsevier.

[11] Massacci, F. Strongly analytic tableaux for normal modal logics. In *Proceedings of the 12th International Conference on Automated Deduction (CADE'94)* (1994), A. Bundy, Ed., vol. 814, Springer-Verlag, pp. 723–737.

[12] Spaan, E. *Complexity of Modal Logics*. PhD thesis, ILLC, University of Amsterdam, 1993.