Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 174 (2007) 15–27
www.elsevier.com/locate/entcs

Higher-Order Syntax and Saturation Algorithms for Hybrid Logic

Moritz Hardt[1] Gert Smolka[1]

Programming Systems Lab
Saarland University
Saarbrücken, Germany

Abstract

We present modal logic on the basis of the simply typed lambda calculus with a system of equational deduction. Combining first-order quantification and higher-order syntax, we can maintain modal reasoning in terms of classical logic by remarkably simple means. Such an approach has been broadly uninvestigated, even though it has notable advantages, especially in the case of Hybrid Logic.
We develop a tableau-like semi-decision procedure and subsequently a decision procedure for an alternative characterization of $\mathcal{HL}(@)$, a well-studied fragment of Hybrid Logic.
With regards to deduction, our calculus simplifies in particular the treatment of identities. Moreover, labeling and access information are both internal and explicit, while in contrast traditional modal tableau calculi either rely on external labeling mechanisms or have to maintain an implicit accessibility relation by equivalent formulas.
With regards to computational complexity, our saturation algorithm is optimal. In particular, this proves the satisfiability problem for $\mathcal{HL}(@)$ to be in PSPACE, a result that was previously not achieved by the saturation approach.

Keywords: Hybrid logic, modal logic, lambda calculus, tableau systems, decision procedures

1 Introduction

When explaining the features of a modal logic, modal logicians stress the point that these languages support an *internal* view on a relational structure, while on the contrary classical logic employs *external* mechanisms such as quantification and variable-binding [4]. Consequently, modal logics deserve special-purpose syntax and semantics which capture this essential idea. Surprisingly though, most texts on modal logic introduce "standard translations", mappings which recursively eliminate modal syntax in favor of first-order predicate logic. This exhibits a trade-off. On the one hand, modal reasoning is comfortable in its traditional presentation. On the other hand, the coherence of classical syntax with standard semantics is desirable.

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

1571-0661
doi:10.1016/j.entcs.2006.11.023

Taking these points into consideration, we develop a formulation of modal logic based on the simply typed lambda calculus [7] with a system of equational deduction [14]. By combining first-order quantification and higher-order syntax, Kripke semantics is fully internalized in our approach. With a few key definitions, syntax, semantics and deduction is set up for modal logics. While we make a commitment to classical logic, we maintain the "local perspective" of modal logic in our native syntax. Even more so, traditional modal syntax is preserved on a notational level. This way, standard translations are obsolete. Along the line, our approach remedies the problems with β-conversion as they appear in the work of Fitting [11,9] who gives examples of β/η-equal modal terms denoting differently, which is clearly in conflict with the lambda calculus. In order to avoid phenomena like these, we make *names* for points of evaluation explicit, already in the case of our minimal modal logic $\mathcal{MF}$.

This leads us to the consideration of Hybrid Logic. In a common formulation, often referred to as $\mathcal{HL}(@,\downarrow)$, Hybrid Logic enriches the minimal modal logic $\mathcal{K}$ with naming, binding, and referencing constructions, as well as an implicit identity judgement [5,1,4]. We offer an equivalent logic called $\mathcal{MFI}$ that extends $\mathcal{MF}$ with the identity predicate. Once $\mathcal{HL}(@)$ is characterized as the *monadic* fragment of $\mathcal{MFI}$, we develop a tableau calculus and a tableau-based decision procedure for this logic. The first such algorithm is due to Tzakova [15]. Recently, Bolander and Braüner [6] extended Tzakova's system by a treatment of the universal modalities and a proof of termination using loop-checking techniques. A semi-decision procedure in form of a tableau calculus is given by Blackburn [3] who discusses the advantages of internalizing labeled deduction in Hybrid Logic. However, none of these results is optimal with respect to computational complexity.

Contributions and Overview

In Section 3, we present syntax, semantics, and deduction for modal logics on the basis of classical logic. Traditional modal syntax is preserved on a notational level, which makes standard translations dispensable. We solve previous problems with β-conversion that are inherent in traditional modal syntax. Proving equivalence, we provide alternative characterizations of $\mathcal{HL}(@,\downarrow)$ and $\mathcal{HL}(@)$.

In Section 4, we develop a tableau-based semi-decision procedure for monadic $\mathcal{MFI}$, i.e., $\mathcal{HL}(@)$. Careful analysis of the role of the identity predicate in monadic $\mathcal{MFI}$ leads to simple saturation rules. Moreover, deduction is fully internal carrying forward [3], but in contrast to [3] we still profit from an explicit access relation at the object level.

In Section 5, we extend our saturation approach to a polynomial space decision procedure for the satisfiability problem of monadic $\mathcal{MFI}$. We are not aware of any previous tableau-based result that matches the PSPACE lower bound for this problem.

Extended discussion of the sections 3 and 4 can be found in [13].

2 The Logical Base

We consider a simply typed lambda calculus where every term has a unique type. Interpretation for terms and types is provided by standard semantics. In our system, logical constants are axiomatized by equations and deduction itself is purely equational. For a full discussion of this topic see [14].

Figure 1 specifies a system of first-order predicate logic which includes derived modal operators. The axioms are known to enforce a canonical interpretation of the constants.

Theory	ML	
Base Types	B, V	
Constants	$0 : \mathsf{B}$	
	$\to : \mathsf{B} \to \mathsf{B} \to \mathsf{B}$	
	$\forall : (\mathsf{V} \to \mathsf{B}) \to \mathsf{B}$	
	$\doteq : \mathsf{V} \to \mathsf{V} \to \mathsf{B}$	
	$R : \mathsf{V} \to \mathsf{V} \to \mathsf{B}$	
Axioms	$0 \to x = 1$	
	$1 \to x = x$	
	$x \vee y = y \vee x$	Commutativity
	$f0 \to f1 \to fx = 1$	Boolean Case Analysis
	$\forall(\lambda x.1) = 1$	
	$\forall f \to fx = 1$	Instantiation
	$x \doteq x = 1$	Reflexivity
	$x \doteq y \to fx \to fy = 1$	Replacement
Derived Constants	$1 = 0 \to 0$	
	$\neg x = x \to 0$	
	$x \vee y = (x \to y) \to y$	
	$x \wedge y = \neg(\neg x \vee \neg y)$	
	$\exists f = \neg(\forall(\lambda x.\neg(fx)))$	
	$\Box xf = \forall y.Rxy \to fy$	
	$\Diamond xf = \exists y.Rxy \wedge fy$	
Notation	$\forall x.t = \forall(\lambda x.t)$	
	$\exists x.t = \exists(\lambda x.t)$	

Fig. 1. First-order Predicate Logic with Modal Operators

For each type we assume a countably infinite set of variables. For a term t, $\mathsf{FV}\,t$ denotes the set of free variables of type V occurring in t. If $\mathsf{FV}\,t = \emptyset$, we say that t is *closed*. We furthermore assume a countably infinite set Par of constants of type V called *parameters*. The set of parameters occurring in a term t is denoted by $\mathsf{Par}\,t$.

Variables of type B and V are written as x, y, while we use a, b for parameters. The letters u, v may either refer to a variable of type V or a parameter.

We denote by $t[u := v]$ the term obtained from t by replacing all (capture-free) occurrences of u by v. The *size* of a term t, denoted by $|t|$, is defined by structural recursion as usual. A *formula* is a term of type B.

3 Modal Logic Revisited

Our presentation of modal logic relies on the following key components.

- Type constant V
- Names $u, v : \mathsf{V}$
- Propositional variables $f, g : \mathsf{V} \to \mathsf{B}$
- Relational constant $R : \mathsf{V} \to \mathsf{V} \to \mathsf{B}$
- Modal operators $\Diamond, \Box : \mathsf{V} \to (\mathsf{V} \to \mathsf{B}) \to \mathsf{B}$

The type constant V is interpreted by a non-empty set of objects called *vertices*. Propositional variables denote predicates on vertices. Accordingly, the modal operators enable judgments about a *vertex* and a *property of vertices*. It is straightforward to pin down the semantics of the modal operators by means of the two axioms depicted in Figure 1. Arranged in a more suggestive way, these terms enjoy the intuitive reading:

$\Box u(\lambda x.t)$ "At u, all direct successors x satisfy t."

$\Diamond u(\lambda x.t)$ "At u, some direct successor x satisfies t."

Surprisingly, a "traditional" modal syntax is still available on a notational level. For this purpose, we reserve a single fixed variable $\pi : \mathsf{V}$ and think of π as the current point of evaluation in our model. This is the standard technique to mimic the external positional argument in Kripke semantics. In a next step, we *situate* propositional variables as well as the modal operators at π, and hide the free variable by the notation introduced in Figure 2. This allows us to define a traditional minimal modal logic as follows.

Definition 3.1 $t \in \mathcal{K} ::= \mathring{f} \mid \neg t \mid t \wedge t \mid \mathring{\Box} t$

By dropping the restriction to a single name, we arrive at the first *modal fragment* of our choice.

Definition 3.2 $t \in \mathcal{MF} ::= fu \mid \neg t \mid t \wedge t \mid \Box u(\lambda x.t)$

$\mathcal{MF}$ subsumes our modal notation. On the other hand, there are certain formulas which lack a single-variable equivalent, e.g., $fa \wedge \neg(fb)$. The point is that $\mathcal{MF}$ already is *hybrid* in that it delivers certain *naming* and *binding* capabilities which are absent in $\mathcal{K}$.

Notation	Definition	Remark
	π	fixed variable, current point of evaluation
$\mathring{f}$	$f\pi$	
$\mathring{\Box}t$	$\Box\pi(\lambda\pi.t)$	
$\mathring{u}$	$\pi \doteq u$	
$@u.t$	$(\lambda\pi.t)u$	occurrences of π captured and replaced by u
$\downarrow x.t$	$(\lambda x.t)\pi$	introduces x as a name for π

Fig. 2. Modal and Hybrid Notation

3.1 Hybrid Logic, Identity and $\mathcal{MFI}$

Hybrid Logic is an extension of $\mathcal{K}$ with the advantage that it allows to *name* and *identify* vertices. For this purpose, $\mathcal{HL}$ introduces three new constructions as depicted in Figure 2.

Definition 3.3 $t \in \mathcal{HL} ::= \mathring{f} \mid \mathring{u} \mid \neg t \mid t \wedge t \mid \mathring{\Box}t \mid @u.t \mid \downarrow x.t$

A formula $\mathring{u}$ is called a *nominal*, @ the *satisfaction*-operator, and $\downarrow$ the *down*-operator. $\mathcal{HL}$ is often referred to as $\mathcal{HL}(@,\downarrow)$ to distinguish the fragment $\mathcal{HL}(@)$ which excludes the down-operator. In this case, $u \in \mathsf{Par}$, by convention. Since the remaining free variable π may be replaced by a fresh parameter with the help of the satisfaction-operator, we assume without loss of generality that terms in $\mathcal{HL}(@)$ are closed.

Via β-reduction, $\mathcal{HL}$ maps into the following extension of $\mathcal{MF}$.

Definition 3.4 $t \in \mathcal{MFI} ::= fu \mid u \doteq v \mid \neg t \mid t \wedge t \mid \Box u(\lambda x.t)$

Interestingly, this time there exists an inverse mapping. While $\mathcal{K}$ and $\mathcal{MF}$ do not match up, we find essentially for every term in $\mathcal{MFI}$ an equivalent one in modal notation.

Proposition 3.5 *Consider the following mapping $\varphi \in \mathcal{MFI} \to \mathcal{HL}$ defined by recursion on $t \in \mathcal{MFI}$. It holds for all $t \in \mathcal{MFI}$ with $\pi \notin \mathsf{FV}\, t$ that $\mathsf{ML} \vdash t = \varphi t$.*

$$\begin{aligned} \varphi(fu) &= @u.\mathring{f} \\ \varphi(\neg t) &= \neg(\varphi t) \\ \varphi(t \wedge t') &= (\varphi t) \wedge (\varphi t') \\ \varphi(u \doteq v) &= @u.\mathring{v} \\ \varphi(\Box u(\lambda x.t)) &= @u.\mathring{\Box}(\downarrow x.\varphi t) \end{aligned}$$

As a matter of fact, $\mathcal{HL}$ and $\mathcal{MFI}$ coincide in a natural way. But, is there a similar result for $\mathcal{HL}(@)$? The operators $\downarrow$ and @ are eliminated by β-reduction and therefore indistinguishable in $\mathcal{MFI}$. There is no operator we could simply omit. Instead, we must find a different characterization of $\mathcal{HL}(@)$.

Considering nominals $\mathring{a}$, we find that each identity in $\mathcal{HL}(@)$ contains a parameter. We call this property *quasi-monadicity*.

Definition 3.6 (Quasi-Monadic Formula) We call a formula $t \in \mathcal{MFI}$ *quasi-*

monadic, if every subterm of the form $u \doteq v$ contains a parameter.

Moreover, the scope of the quantifiers is well-nested. That is, every term can be represented with a single bound variable.

Definition 3.7 (Monadic Formula) A formula $t \in \mathcal{MFI}$ is called *monadic*, if it is quasi-monadic and every subterm of the form $\lambda x.t'$ is closed.

Definition 3.8 $\mathcal{MFI}_1 \stackrel{def}{=} \{t \in \mathcal{MFI} \mid t \text{ monadic and closed}\}$

It is straightforward to prove the analogon of Proposition 3.5, that is, to define a mapping $\varphi \in \mathcal{MFI}_1 \to \mathcal{HL}(@)$ such that for all $t \in \mathcal{MFI}$, it holds $\mathsf{ML} \vdash t = \varphi t$. Proof of this fact and the following proposition is found in [13].

Proposition 3.9 *For every quasi-monadic formula, we can compute an equivalent monadic formula.*

3.2 The Case of Modal Base Syntax

Let us evaluate modal notation as a possible base syntax. More precisely, we consider $\mathring{\Box}$ a constant of type $\mathsf{B} \to \mathsf{B}$, $\mathring{f}$ a constant of type B and so on. Now, with respect to modal semantics, the terms $t_1 = \mathring{\Box}\mathring{f}$ and $t_2 = (\lambda q.\mathring{\Box}q)\mathring{f}$ must denote differently, if $\mathring{f}$ is a "non-rigid" constant. For that reason we should be able to distinguish these formulas. However, $\lambda q.\mathring{\Box}q$ η-reduces in one step to $\mathring{\Box}$, and t_2 even β-equals t_1. So, with respect to the underlying lambda calculus, there is absolutely no justification to discriminate t_1 and t_2.

Surprisingly, on a notational level these terms are uncritical. The formula t_2 is a short hand for $(\lambda q.\Box\pi(\lambda\pi.q))(f\pi)$. As substitution does *not* capture, this term β-reduces to $\Box\pi(\lambda\pi'.f\pi)$ in contrast to $t_1 = \Box\pi(\lambda\pi.f\pi)$. In fact, analyzing t_2 deductively in ML yields the modal formula with the intended semantics.

$$t_2 = \Box\pi(\lambda\pi'.f\pi) = \Box\pi(\lambda\pi'.0 \vee f\pi) = \Box\pi(\lambda\pi'.0) \vee f\pi = \mathring{\Box}0 \vee \mathring{f}$$

The third equation can be derived from well-known quantifier laws. Remarkably, by classical reasoning, we solve a problem related to modal syntax.

4 A Saturation Procedure for $\mathcal{MFI}_1$

In this section, we devise a tableau-like semi-decision procedure for $\mathcal{MFI}_1$. In a way that will be suitable for our later analysis, we restate a version of $\mathcal{MFI}$ in negation normal form as well as slightly modified syntactic characterizations.

Definition 4.1 $\mathcal{MFI}$ is the set of formulas of the following form.

$$t ::= fu \mid \neg(fu) \mid u \doteq v \mid \neg(u \doteq v) \mid Ruv \mid t \wedge t \mid t \vee t \mid \Box u(\lambda x.t) \mid \Diamond u(\lambda x.t)$$

Definition 4.2 A formula $t \in \mathcal{MFI}$ is called

- *proper*, if it does not contain any subterm of the form Ruv.
- *quasi-monadic*, if it is proper and every subterm of the form $u \doteq v$ contains a parameter.

- *monadic*, if it is quasi-monadic, $|\mathsf{FV}\, t| \leq 1$ and every subterm of the form $\lambda x.t'$ is closed.

Definition 4.3 $\mathcal{MFI}_1 \stackrel{def}{=} \{t \in \mathcal{MFI} \mid t \textit{ monadic and closed}\}$

Definition 4.4 (Trivial) A set of formulas C is called *trivial* if either $\{t, \neg t\} \subseteq C$ or $\neg(t \doteq t) \in C$ for some term t.

Definition 4.5 (Purely Monadic) A set of formulas C is *purely monadic*, if $C \subseteq \mathcal{MFI}_1$. It is *monadic*, if all members are either monadic or of the form Ruv.

Definition 4.6 A set of formulas is *satisfiable* if and only if there exists an interpretation $\mathcal{I}$ and an assignment σ such that $\mathcal{I} \models \mathsf{ML}$ and $\mathcal{I}, \sigma \models t = 1$ for every $t \in C$

Definition 4.7 (Clause) A *clause* is a finite set of formulas. The *degree* of an empty clause is 0, otherwise $\deg C \stackrel{\text{def}}{=} \max_{t \in C} |t|$.

Definition 4.8 We will use the notation $(\lambda x.t)\downarrow u \stackrel{\text{def}}{=} t[x := u]$.

4.1 Saturatedness

The design space for our later calculus will be given in this section in terms of saturatedness conditions. The idea is that if a set of formulas respects these closure conditions and we cannot observe an obvious contradiction, then, in fact, this set must be satisfiable. We will formulate this result as a model existence theorem.

Definition 4.9 (Saturatedness) A set of formulas C is *saturated*, if it satisfies all of the following conditions.

($\mathcal{S}_c$) C is not trivial

($\mathcal{S}_\wedge$) If $s \wedge t \in C$, then $\{s, t\} \subseteq C$.

($\mathcal{S}_\vee$) If $s \vee t \in C$, then $s \in C$ or $t \in C$.

($\mathcal{S}_\Diamond$) If $\Diamond ut \in C$, then $\{Rux, t\downarrow x\} \subseteq C$ for some x.

($\mathcal{S}_\Box$) If $\Box ut \in C$ and $Ruv \in C$, then $t\downarrow v \in C$.

($\mathcal{S}^s_{\doteq}$) If $u \doteq v \in C$, then $v \doteq u \in C$.

($\mathcal{S}_{\doteq}$) If $u \doteq a \in C$ and $t \in C$, then $t[u := a] \in C$.

Definition 4.10 For a set of formulas C, we define $\sim_C$ to be the least equivalence relation such that $u \sim_C v$ whenever $u \doteq v \in C$. We write $[u]_C \stackrel{\text{def}}{=} \{v \mid u \sim_C v\}$ to denote the *equivalence class* of u with respect to $\sim_C$. An equivalence class $[u]_C$ is called *trivial*, if $[u]_C = \{u\}$.

Proposition 4.11 (Parameter Existence) *Let C be a monadic set of formulas. Every nontrivial equivalence class $[u]_C$ contains a parameter.*

Proposition 4.12 (Agreement) *Let C be a saturated monadic set of formulas. Given $u \neq a$, it holds $u \sim_C a$ if and only if $u \doteq a \in C$.*

A proof of the following result can be found in [13].

Theorem 4.13 (Model Existence) *Every (finite) saturated monadic set of formulas is satisfiable in a (finite) model.*

Proof (Sketch) Given a saturated monadic set of formulas C, our goal is to construct a model satisfying C. The interpretation of V will be the set of all equivalence classes of $\sim_C$. To connect this domain of our model with the terms in C, we need representatives. The crucial idea is to choose a parameter as the representative of a nontrivial equivalence class (cf. Parameter Existence). Having done so, we can use Agreement (Proposition 4.12) and $(\mathcal{S}_{\doteq})$ to argue that this parameter holds a copy of the appropriate terms. □

4.2 Saturation

Previously, we established the notion of a saturated monadic set of formulas and proved that such sets are satisfiable. The computational counterpart is a procedure which performs saturation steps. We approach this goal by defining a binary relation between clauses by means of a few easily computable rules.

Definition 4.14 (Saturation) We let γ be a mapping from clauses to variables of type V such that $\gamma C \notin \mathsf{FV}\, C$. The saturation relation $\to$ over clauses is defined as follows:

$C \to D$ if and only if $C \subset D$ and D can be obtained from C by applying one of the following rules.

$(\mathcal{C}_\wedge)$ 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 $\Diamond ut \in C$, add $Ru(\gamma C)$ and $t{\downarrow}(\gamma C)$.

$(\mathcal{C}_\Box)$ If $\Box ut \in C$ and $Ruv \in C$, add $t{\downarrow}v$.

$(\mathcal{C}^s_{\doteq})$ If $u \doteq v \in C$, add $v \doteq u$.

$(\mathcal{C}_{\doteq})$ If $u \doteq a \in C$ and $t \in C$, add $t[u := a]$.

We say $C \to D$ *don't care*, if $C \to D$ by one of the saturation rules excluding $(\mathcal{C}_\vee)$. We say $C \to D_1, D_2$ *don't know*, if D_1 and D_2 are the two alternative results of applying $(\mathcal{C}_\vee)$ to some $s \vee t \in C$.

It is a simple task to show that these rules satisfy the key properties *Soundness* and *Completeness*. Soundness ensures that satisfiability propagates back and forth over the application of a saturation rule.

Proposition 4.15 (Soundness) (i) *If $C \to D$ don't care, then C is satisfiable if and only if D is satisfiable.*

(ii) *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.*

To formulate completeness, we will call a purely monadic clause C *consistent*, if from C we cannot derive a conflict. More precisely, there exists no clause D with $C \subseteq D$ such that D can be obtained from C by applying any finite sequence of saturation steps.

Proposition 4.16 (Completeness) *Consistent purely monadic clauses are satisfiable.*

Proof. Let C be a consistent purely monadic clause. We can apply the saturation rules in a systematic way in order to obtain a (not necessarily finite) set of formulas D with $C \subseteq D$ such that D cannot be extended by application of a saturation rule. This is a standard technique as described in, for example, [11,3]. Since C is consistent, D is not trivial. Moreover, D is monadic, since monadicity is preserved by saturation [13].

Consequently, C is a saturated monadic set of formulas and thus satisfiable by our previous model existence theorem. □

4.3 Related Tableau Calculi

Tzakova [15] was the first to introduce a tableau-based decision procedure for $\mathcal{HL}(@)$. Besides the standard modal and boolean rules, her system comprises four rules concerning @ and nominals (identity). In order to achieve termination, Tzakova states an involved special-purpose branch extension procedure. Shortly after, Blackburn [3] discusses the advantages of internalizing labeled deduction for Hybrid Logic. He stresses the point that nominals should be considered formulas in order to establish labeling discipline at the object level. To handle identities, Blackburn introduces four rules: Reflexivity, Symmetry, Replacement and a rule called "Bridge". Finally, Bolander and Bräuner [6] extend the calculi of Tzakova and Blackburn by a treatment of the universal modalities and give a simplified discussion. In the case of Tzakova's system, they recognize that two of her rules are subsumed by a strong replacement rule as used by Blackburn. However, such a strong replacement rule immediately requires loop-checking as shown in [13].

Our calculus carries forward the arguments of Blackburn, as state labels are integral parts of formulas. Blackburn avoids the use of meta-level information and thus represents the successor relation Ruv by an equivalent formula like $\Diamond u(\lambda x.x \doteq v)$ or $@u.\mathring{\Diamond}\mathring{v}$, respectively. To maintain this representation, he must install the additional rule "Bridge".

$$\frac{@\mathring{u}.\mathring{\Diamond}\mathring{v} \quad @\mathring{v}.\mathring{a}}{@u.\mathring{\Diamond}\mathring{a}}$$

In our system, the access relation itself is a formula. But then, "Bridge" is just a special case of $(\mathcal{C}_{\doteq})$: $\{Ruv, v \doteq a\} \to \{\ldots, Rua\}$.

In contrast to Blackburn's system, many modal tableaux as those of Fitting [8,10] and Gabbay [12] maintain external state labels and access information. It turns out that both of these can be naturally represented at the object level.

5 Saturation in Polynomial Space

In this section we describe a decision procedure for the satisfiability problem of $\mathcal{MFI}_1$ based on our saturation rules. A central feature of this algorithm is that it matches the PSPACE lower bound for $\mathcal{HL}(@)$ [2].

In the case of $\mathcal{MF}$, that is in the absence of identities, a simple standard technique yields an optimal tableau-based decision procedure. We describe this technique as *depth-first saturation*: When a clause C does not admit any further application of the saturation rules except for $(\mathcal{C}_\Diamond)$, we pick a diamond $\Diamond ut \in C$, gather all boxes $\Box ut_1, \ldots, \Box ut_k$ and start saturating the clause $\{t \downarrow x, t_1 \downarrow x, \ldots, t_k \downarrow x\}$. If no contradiction occurs, we continue with $C \backslash \{\Diamond ut\}$.

The intuitive reason why this algorithm uses only polynomial space is that the possibly large set of "edges" Ruv is not stored explicitly. Instead, this graph is traversed "manually" in a depth-first manner. However, in the case of $\mathcal{MFI}$ (even $\mathcal{MF}$) this procedure is without modification incomplete. Closed terms might occur which would have been required at an earlier stage of the algorithm in order to reveal a contradiction.

We can anticipate this problem as follows.

(i) On input of a monadic clause, we guess an equivalence relation on its parameters. With fixed representatives for the equivalence classes, we "simplify" the clause in accordance with our choice. This will decrease the size of the resulting clause and it will prevent us from storing terms several times for related parameters.

(ii) Additionally, we guess which subterms occurring in this updated clause will (for example by means of $(\mathcal{C}_{\doteq})$) eventually be added to the parameter component, i.e., the sub-clause consisting of only closed terms.

(iii) We perform the standard depth-first saturation with respect to these additional information and verify our choice.

To make these points precise, we introduce the following notation.

Definition 5.1 Let C be a monadic clause.

(i) For each equivalence relation on $\mathsf{Par}\, C$ we consider a fixed system of representatives $\rho \in \mathsf{Par}\, C \to \mathsf{Par}\, C$. We write ρt to denote the term obtained from t by replacing each occurring parameter a with its representative ρa.
Accordingly, $\rho C \stackrel{\text{def}}{=} \{\rho t \mid t \in C\}$.

(ii) We define $\mathsf{Clo}\, C$ to be the clause containing all subterms of C where the possible free variable has been replaced by a parameter $a \in \mathsf{Par}\, C$.

(iii) For $\Diamond ut \in C$, we define $C_{\Diamond ut} \stackrel{\text{def}}{=} \{t \downarrow x, t_1 \downarrow x, \ldots, t_k \downarrow x\}$ where $\Box ut_1, \ldots, \Box ut_k$ are all terms in C of the form $\Box ut'$. Furthermore, x is a fresh variable, which we call the *characteristic* variable of the clause $C_{\Diamond ut}$.

Figure 3 introduces our saturation algorithm.

Lemma 5.2 *The saturation algorithm terminates on input of a purely monadic clause C and uses polynomial space.*

Proof. Let C be a purely monadic clause and let us consider $|C| \cdot (\deg C)$ as the input size.

Observe that that each choice of ρ and B in (1) is of polynomial size. In par-

1.	$\mathcal{A}C = \bigvee_\rho \bigvee_B \mathcal{A}_B(B \cup \rho C)$	where $B \subseteq \mathsf{Clo}(\rho C)$
2.	$\mathcal{A}_B C = \mathsf{false}$	if C is trivial or conflicting†
3.	$\mathcal{A}_B C = \mathcal{A}_B D$	if $C \to D$ don't care, excluding $(\mathcal{C}_\diamond)$
4.	$\mathcal{A}_B C = \mathcal{A}_B D_1 \bigvee \mathcal{A}_B D_2$	if $C \to D_1, D_2$ don't know
5.	$\mathcal{A}_B C = \bigwedge_{\diamond ut \in C} \mathcal{A}_B C_{\diamond ut}$	otherwise

$^\dagger C$ is *conflicting*, if $a \doteq b \in C$ and $a \neq b$ or there is a closed term $t \in (C \backslash B)$.

Fig. 3. Saturation Algorithm

ticular, there are finitely many such choices. Moreover, consecutive calls to (3) and (4) have a polynomial bound as we have ruled out $(\mathcal{C}_\diamond)$. On the other hand, whenever we execute (5) on a clause $D \subseteq C$ we know that for every $\diamond ut \in D$, we have $\deg D > \deg D_{\diamond ut}$. Thus, the depth of such recursive calls is bounded in the degree of the input clause C.

If we reuse space in (1), (4), and (5) we achieve an overall space consumption which is bounded by a polynomial in the size of the input. □

Theorem 5.3 *By means of the saturation algorithm we can decide in polynomial space whether or not a purely monadic clause is satisfiable.*

Proof. Let C be a purely monadic clause. After the previous lemma, it remains to prove that C is satisfiable if and only if $\mathcal{A}C = \mathsf{true}$.

If C is satisfiable, there exists a clause D which is a saturated extension of C. From D we obtain appropriate ρ and B for which it is simple to show that $\mathcal{A}_B(B \cup \rho C) = \mathsf{true}$. Consequently, $\mathcal{A}C = \mathsf{true}$.

Conversely, from the fact that $\mathcal{A}C = \mathsf{true}$ we construct a saturated extension of C. That is, if $\mathcal{A}C = \mathsf{true}$, then $\mathcal{A}_B(B \cup \rho C) = \mathsf{true}$ for some ρ and B. By traversing the recursion tree for this procedure call, we obtain clauses $D^1, \ldots, D^m$ where each of these clauses D^i has the property that $\mathcal{A}_B D^i = \bigwedge_{\diamond ut \in D^i} \mathcal{A}_B D^i_{\diamond ut} = \mathsf{true}$, and each D^i was computed by executing $\mathcal{A}_B D^j$ for some $j < i$ where we let $D^0 = B \cup \rho C$. We assume that all characteristic variables are disjoint.

Furthermore, let R be the set of terms Rux whenever x is the the characteristic variable of a clause D^j that was obtained by executing $\mathcal{A}_B D^i_{\diamond ut}$ for some $\diamond ut \in D^i$ and $i < j$. We must also add Rua in case $x \doteq a \in D^j$ for some parameter a.

Finally, we define D to be the clause obtained from $D' = \left(\bigcup_i D^i\right) \cup R$, by adding for each formula $t \in D'$ all those copies of t where the parameters $a_1, \ldots, a_k$ occurring in t have been replaced by arbitrary $b_1, \ldots, b_k$ with $\rho b_i = a_i$.

Clearly, $C \subseteq D$. Moreover, we can prove that D is saturated and thus satisfiable by Model Existence (Theorem 4.13).

$(\mathcal{S}_c)$ Arguing by contradiction, assume D is trivial and $\{t, \neg t\} \subseteq D$. Then, we have $\{\rho t, \neg(\rho t)\} \subseteq D'$ where $\rho t \in D^i$ and $\neg(\rho t) \in D^j$ for some i, j. If ρt is closed, then $\{\rho t, \neg(\rho t)\} \subseteq B$, since neither D^i nor D^j is conflicting. But then, $\mathcal{A}_B(B \cup \rho C) = \mathsf{false}$. If ρt contains a free variable x, then x is the characteristic

variable of both D^i and D^j. Thus, $i = j$ and $\mathcal{A}_B D^i = \mathsf{false}$ follows. Either way it is a contradiction. The case of $\neg(t \doteq t) \in D$ is clear.

$(\mathcal{S}_\wedge), (\mathcal{S}_\vee), (\mathcal{S}^s_{\doteq})$ Straightforward.

$(\mathcal{S}_\Diamond)$ Assume $\Diamond ut \in D$, then $\Diamond(\rho u)(\rho t) \in D^i$ for some i. Then, there is D^j with $j > i$ and a characteristic variable x such that $R(\rho u)x \in R$ and $(\rho t)\downarrow x \in D^j \subseteq D'$. Consequently, $\{Rux, t\downarrow x\} \subseteq D$.

$(\mathcal{S}_\Box)$ Suppose $\{\Box ut, Ruv\} \subseteq D$. We demonstrate the case where u is a variable, but v is a parameter. We have $\Box u(\rho t) \in D^i$. But, to obtain the edge $Ruv \in R$, there must be a clause D^j with $j > i$ and a characteristic variable y such that $t\downarrow y \in D^j$ and $y \doteq v \in D^j$. Thus, by application of $(\mathcal{C}_{\doteq})$, we have $(\rho t)\downarrow v \in D'$ and $\rho t \in D$. The case $u, v \in \mathsf{Par}$ is analogous. The remaining two cases are straightforward.

$(\mathcal{S}^\rho_{\doteq})$ Assume $\{u \doteq a, t\} \subseteq D$. We have $\rho u \doteq \rho a \in D'$ and $\rho t \in D'$. If u is a parameter, then $\rho u = \rho a$, since no D^i is conflicting. But then, $t[u := a] \in D$ follows from the way we constructed D. If $u = x$ and $x \in \mathsf{FV}\, t$, then $\{x = \rho a, \rho t\} \subseteq D^i$ where x is the characteristic variable of D^i. By application of $(\mathcal{C}_{\doteq})$, $(\rho t)[x := \rho a] \in D^i \subseteq D'$ and hence, $t[x := a] \in D$.

□

6 Conclusion and Future Work

We presented modal logic on the basis of the simply typed lambda calculus. Our focus was to give modal logic a uniform and natural treatment in terms of classical logic. On the one hand, first-order quantification was strong enough to express the semantics of the modal operators, on the other hand, first-order predicate logic as such was *syntactically too weak* for our purposes. This is why *higher-order syntax* came to play such an important role. We employed higher-order variables, derived higher-order constants and finally expressed operators of Hybrid Logic by means of λ-abstraction. We eventually arrived at three different levels of reasoning:

Notational Level On a notational level, we preserved the traditional syntax of modal logics as in $\mathcal{K}$ and $\mathcal{HL}$.

Native Modal Syntax It turned out that modal notation was naturally subsumed and explained by our native syntax and the fragments $\mathcal{MF}$ and $\mathcal{MFI}$ which we defined in terms of this syntax. However, in the case of $\mathcal{HL}$ we obtained a tight equivalence of notation and syntax.

Quantifiers At the bottom, quantifiers were employed to give modal operators their precise meaning. Validities of modal logic could be derived by equational deduction.

From our point of view, traditional studies in correspondence between modal and first-order predicate logic suffered from their syntactical weakness.

When designing the tableau calculus for $\mathcal{MFI}_1$, our syntax proved to be the appropriate data structure without the need of modification. Both deductively and later with regards to computational matters, the analysis of our procedure

notably seized upon the rich object-level in our system. Although our tamed rule of replacement gives insight into the limited power of the identity predicate in $\mathcal{MFI}_1$, the question remains open whether one can achieve local termination criteria for such a calculus. In the literature, e.g., [6], the argumentation is often that as soon as identities are involved, one faces the same problems as in $\mathcal{K}$ over transitive frames where formulas have to be passed along a chain of successors.

With ideas similar to those in [2], the analysis of identities was also crucial to arrive at a space efficient formulation of our algorithm. It was interesting to see that the design space given by the saturation conditions allowed for nontrivial modifications of our saturation procedure.

We are interested in refinements of our saturation algorithm that make it more practical. Techniques, such as "lazy-guessing", are required to avoid the large computational overhead caused by the initial guessing.

Acknowledgments

We thank an anonymous referee for pointing out a flaw in a previous version of the paper. We appreciate the time the editors gave us for reorganizing the results.

References

[1] Areces, C., P. Blackburn and M. Marx, *Hybrid logic is the bounded fragment of first order logic*, in: *Proceedings of 6th Workshop on Logic, Language, Information and Computation*, 1999, pp. 33–50.

[2] Areces, C., P. Blackburn and M. Marx, *A road-map on complexity for hybrid logics*, in: J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, number 1683 in LNCS (1999), pp. 307–321.

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

[4] Blackburn, P., M. de Rijke and Y. Venema, "Modal Logic," Cambridge Tracks in Theoretical Computer Science, Cambridge University Press, 2001.

[5] Blackburn, P. and J. Seligman, *What are hybrid languages?*, , **1**, CSLI Publications, Stanford University, 1998 pp. 41–62.

[6] Bolander, T. and T. Braüner, "Two Tableau-Based Decision Procedures for Hybrid Logic," Informatik-Berichte **194**, 4th Workshop: Methods for Modalities, Proceedings, 2005, 79-96 pp.

[7] Church, A., *A formulation of the simple theory of types*, Journal of Symbolic Logic **5** (1940), pp. 56–68.

[8] Fitting, M., "Proof Methods for Modal and Intuitionistic Logics," D. Reidel Publishing Co., Dordrecht, 1983.

[9] Fitting, M., *Higher-order modal logic – a sketch* (2001).

[10] Fitting, M., "Types, tableaus, and Gödel's god," Trends in Logic : Studia Logica Library **012**, Kluwer Academic Publishers, 2002.

[11] Fitting, M. and R. L. Mendelsohn, "First-Order Modal Logic," Kluwer Academic Publishers, 1998.

[12] Gabbay, D., "Labelled Deductive Systems," Oxford University Press, 1996.

[13] Hardt, M., "Bachelor's Thesis: Hybrid Logic Revisited," Saarland University, 2006.
URL http://www.ps.uni-sb.de/~hardt/hlrev.html

[14] Smolka, G., "Lecture Notes: Introduction to Computational Logic," Saarland University, 2006.
URL http://www.ps.uni-sb.de/courses/cl-ss06/script/index.html

[15] Tzakova, M., *Tableau calculi for hybrid logics*, in: N. V. Murray, editor, *Analytic Tableaux and Related Methods, TABLEAUX'99*, LNAI **1617** (1999), pp. 278–292.