On the Complexity of Hybrid Logics with Binders

Balder ten Cate[1] and Massimo Franceschet[1,2]

[1] Informatics Institute, University of Amsterdam,
Kruislaan 403 – 1098 SJ Amsterdam, The Netherlands
[2] Department of Sciences, University "G. D'Annunzio",
Viale Pindaro, 42 – 65127 Pescara, Italy

Abstract. *Hybrid logic* refers to a group of logics lying between modal and first-order logic in which one can refer to individual states of the Kripke structure. In particular, the hybrid logic HL($@, \downarrow$) is an appealing extension of modal logic that allows one to refer to a state by means of the given names and to dynamically create new names for a state.
Unfortunately, as for the richer first-order logic, satisfiability for the hybrid logic HL($@, \downarrow$) is undecidable and model checking for HL($@, \downarrow$) is PSPACE-complete. We carefully analyze these results and we isolate large fragments of HL($@, \downarrow$) for which satisfiability is decidable and model checking is below PSPACE.

1 Introduction

There is a general interest in well-behaved logical languages in-between the basic modal language and full first-order logic. Ideally, one would like such languages to combine the good properties of both: to be reasonably expressive, to be decidable, and to have other good properties, such as the interpolation property. Famous examples of fragments that have been studied are the *guarded fragment* [1, 2] and the *two variable fragment* [3, 4]. Both are decidable, reasonably expressive languages, but they lack interpolation.

The hybrid logic HL($@, \downarrow$) is another example of a language in between the basic modal language and full first-order logic. It extends the basic modal language with three constructs: nominals, which act as names of states of the model, the satisfaction operator @, which allows one to express that a formula holds at the state named by a nominal, and the binder $\downarrow$, which allows one to give a name to the current state. Together, these three elements greatly increase the expressivity of the language. Moreover, like the basic modal language and full first-order logic, HL($@, \downarrow$) has the interpolation property. Unfortunately, it is undecidable.

The language HL($@, \downarrow$) is a natural fragment of first-order logic: it is the generated submodel invariant fragment [5], it is the least expressive extension of the basic hybrid language HL(@) with interpolation [6], and, finally, it has been characterized as the intersection of first-order logic with second-order propositional modal logic [7]. HL($@, \downarrow$) has been used in the context of semistructured

data. In particular, [8] gives an application of model checking in hybrid logic to the problems of query and constraint evaluation for semistructured data.

In this paper, we give an in-depth analysis of the undecidability of HL($@, \downarrow$). We show how decidability can be regained by making a syntactic restriction on the formulas, or by restricting the class of models in a natural way. Moreover, we show how these and similar syntactic and semantic restrictions affect the complexity of the model checking problem for hybrid languages.

In Section 2 we introduce hybrid logic, and in Section 3 we revisit the undecidability result for HL($@, \downarrow$). In Section 4 and 5, we show how decidability can be regained by restricting the language and the class of models, respectively. In Section 6 we investigate how these and similar restrictions affect the complexity of the model checking problem for hybrid logic. We conclude in Section 7.

2 Hybrid Logic

In its basic version, hybrid logic extends modal logic with devices for naming (individual) states and for accessing states by their names. The key idea is the use of *nominals*. Syntactically, nominals behave like ordinary propositions, but they have an important semantic property. A nominal is true at *exactly one state* of the model. In such a way, it gives a name to that point. Besides nominals, the hybrid language HL($@, \downarrow$) also contains @-operators, that allow one to state that a formula is true at a state named by a nominal, and the $\downarrow$-binder, that allows one to introduce variables to name points. Formally, HL($@, \downarrow$) is defined as follows.

Let PROP $= \{p, q, \ldots\}$ be a (countably) infinite set of proposition symbols, NOM $= \{i, j, \ldots\}$ be a (countably) infinite set of nominals, and SVAR $= \{x, y, \ldots\}$ be a (countably) infinite set of state variables. We assume that these sets are disjoint. The formulas of the hybrid language HL($@, \downarrow$) are given by the following recursive definition.

$$\alpha := \top \mid p \mid t \mid \neg\alpha \mid \alpha \wedge \beta \mid \Diamond\alpha \mid @_t\alpha \mid \downarrow x.\alpha$$

where $p \in$ PROP, $t \in$ NOM $\cup$ SVAR and $x \in$ SVAR. We will use the familiar shorthand notations, such as $\Box\alpha$ for $\neg\Diamond\neg\alpha$. The notions of *free* and *bound* variables are defined similarly as in first-order logic. A hybrid *sentence* is a hybrid formula with no free variables. The *width* of a formula α is the maximum number of free variables of any subformula of α.

The binder $\downarrow$ binds a variable to the current state of evaluation. For instance, the formula $\downarrow x.\Diamond x$ says that the current state is reflexive. The @ operator combines naturally with the $\downarrow$ binder: while $\downarrow$ stores the current state of evaluation, @ enables us to retrieve the information stored by shifting the point of evaluation. As an example, the formula $\downarrow x.\Diamond\downarrow y.@_x\Box y$ states that the current point has exactly one successor.

Hybrid formulas are interpreted over *hybrid Kripke structures* (or *hybrid models*) of the form $M = (W, R, V)$ where W is a set of states, R is a binary relation over W called the accessability relation, and $V :$ PROP$\cup$NOM $\rightarrow \wp(W)$

is a valuation function that assigns to each proposition letter or nominal a set of states, such that $V(i)$ is a singleton set for each nominals i. The pair $F = (W, R)$ is called the *frame* of M and M is said to be a model based on the frame F.

An assignment for M is a function $g : \mathrm{SVAR} \to W$. Given such an assignment g, a variable $x \in \mathrm{SVAR}$ and a state $w \in W$, we will use g^x_w to refer to the assignment that is identical to g except that maps x to w. Formally, $g^x_w(y) = x$ for $y = x$ and $g^x_w(y) = g(y)$ for $y \neq x$.

Let $M = (W, R, V)$ be a hybrid model, g an assignment for M, and let $w \in W$. For any nominal i, let $[i]^{M,g} = V(i)$, and for any state variable x, let $[x]^{M,g} = \{g(x)\}$. The semantics of $\mathrm{HL}(@, \downarrow)$ is as follows:

$$
\begin{array}{ll}
M, g, w \Vdash \top & \\
M, g, w \Vdash p & \text{iff } w \in V(p) \\
M, g, w \Vdash t & \text{iff } w \in [t]^{M,g} \text{ for } t \in \mathrm{NOM} \cup \mathrm{SVAR} \\
M, g, w \Vdash \neg\alpha & \text{iff } M, g, w \nVdash \alpha \\
M, g, w \Vdash \alpha \wedge \beta & \text{iff } M, g, w \Vdash \alpha \text{ and } M, w \Vdash \beta \\
M, g, w \Vdash \Diamond\alpha & \text{iff there is a } w' \in W \text{ such that } wRw' \text{ and } M, g, w' \Vdash \alpha \\
M, g, w \Vdash @_t\alpha & \text{iff } M, g, w' \Vdash \alpha \text{ where } \{w'\} = [t]^{M,g} \\
M, g, w \Vdash \downarrow x.\alpha & \text{iff } M, g^x_w, w \Vdash \alpha
\end{array}
$$

Define the *first-order correspondence language* to be the first-order language with equality that has one binary relation symbol R, a unary relation symbol p for each $p \in \mathrm{PROP}$ and a constant i for each nominal $i \in \mathrm{NOM}$. Every hybrid Kripke structure (W, R, V) can be viewed as a relational structure for the first-order correspondence language: the binary relation symbol R is interpreted by the accessibility relation R, the unary relation symbols p are interpreted by $V(p)$, and each constant i denotes the unique state w such that $V(i) = \{w\}$. Then, the following *Standard Translation*, defined by mutual recursion[3] between two functions ST_x and ST_y, embeds $\mathrm{HL}(@, \downarrow)$ into the first-order correspondence language (where $p \in \mathrm{PROP}$ and $t \in \mathrm{NOM} \cup \mathrm{SVAR}$):[4]

$$
\begin{array}{l|l}
\mathrm{ST}_x(\top) = \top & \mathrm{ST}_y(\top) = \top \\
\mathrm{ST}_x(p) = p(x) & \mathrm{ST}_y(p) = p(y) \\
\mathrm{ST}_x(t) = x = t & \mathrm{ST}_y(t) = y = t \\
\mathrm{ST}_x(\neg\alpha) = \neg\mathrm{ST}_x(\alpha) & \mathrm{ST}_y(\neg\alpha) = \neg\mathrm{ST}_y(\alpha) \\
\mathrm{ST}_x(\alpha \wedge \beta) = \mathrm{ST}_x(\alpha) \wedge \mathrm{ST}_x(\beta) & \mathrm{ST}_y(\alpha \wedge \beta) = \mathrm{ST}_y(\alpha) \wedge \mathrm{ST}_y(\beta) \\
\mathrm{ST}_x(\Diamond\alpha) = \exists y.(xRy \wedge \mathrm{ST}_y(\alpha)) & \mathrm{ST}_y(\Diamond\alpha) = \exists x.(yRx \wedge \mathrm{ST}_x(\alpha)) \\
\mathrm{ST}_x(@_t\alpha) = \exists y.(y = t \wedge \mathrm{ST}_y(\alpha)) & \mathrm{ST}_y(@_t\alpha) = \exists x.(x = t \wedge \mathrm{ST}_x(\alpha)) \\
\mathrm{ST}_x(\downarrow z.\alpha) = \exists z.(z = x \wedge \mathrm{ST}_x(\alpha)) & \mathrm{ST}_y(\downarrow z.\alpha) = \exists z.(z = y \wedge \mathrm{ST}_y(\alpha))
\end{array}
$$

[3] Mutual recursion is used in order to limit the number of variables occurring in the translation.

[4] As was pointed out by Guillaume Malod (personal communication), the clause for the $\downarrow$-binder in the Standard Translation for $\mathrm{HL}(@, \downarrow)$ given in [5], i.e., $\mathrm{ST}_x(\downarrow z.\alpha) = \mathrm{ST}_x(\alpha)[z/x]$ and $\mathrm{ST}_y(\downarrow z.\alpha) = \mathrm{ST}_y(\alpha)[z/y]$, is incorrect. Indeed, consider the formula $\downarrow z.\Diamond\Diamond z$. The Standard Translation of this formula according to the definitions in [5] is $\exists y.(xRy \wedge \exists x.(yRx \wedge x = z))[z/x] = \exists y.(xRy \wedge \exists x.(yRx \wedge x = x))$, which clearly fails to capture the semantics of the hybrid formula.

Here, it is assumed that the variables x, y do not occur in α. For each HL($@, \downarrow$)-formula α with free variables $y_1, \ldots, y_n$, $\mathrm{ST}_x(\alpha)$ is a first-order formula with free variables in $\{x, y_1, \ldots, y_n\}$. Moreover, it is easy to show that for any Kripke structure M, assignment g and world w, $M, g, w \Vdash \alpha$ if, and only if, $M, g^x_w \models \mathrm{ST}_x(\alpha)$. It follows that HL($@, \downarrow$) is a fragment of the first-order correspondence language. In fact, this fragment admits several natural characterizations, as mentioned in the introduction.

In Section 4, we will consider a further extension of HL($@, \downarrow$), containing the global modality E and the converse operator $\Diamond^-$ (whose duals will be denoted by A and $\Box^-$, respectively). These have the following semantics:

$M, g, w \Vdash E\alpha$ iff there is a $w' \in W$ such that $M, g, w' \Vdash \alpha$
$M, g, w \Vdash \Diamond^-\alpha$ iff there is a $w' \in W$ such that $w'Rw$ and $M, g, w' \Vdash \alpha$

or, in terms of the Standard Translation:

$$\begin{array}{l|l} \mathrm{ST}_x(E\alpha) = \exists y.(y = y \wedge \mathrm{ST}_y(\alpha)) & \mathrm{ST}_y(E\alpha) = \exists x.(x = x \wedge \mathrm{ST}_x(\alpha)) \\ \mathrm{ST}_x(\Diamond^-\alpha) = \exists y.(yRx \wedge \mathrm{ST}_y(\alpha)) & \mathrm{ST}_y(\Diamond^-\alpha) = \exists x.(xRy \wedge \mathrm{ST}_x(\alpha)) \end{array}$$

For $\theta_1, \ldots, \theta_n \in \{\downarrow, @, E, \Diamond^-\}$, we will use HL($\theta_1, \ldots, \theta_n$) to refer to the extension of the modal language with nominals and the operators $\theta_1, \ldots, \theta_n$ (if $\downarrow$ is among $\theta_1, \ldots, \theta_n$, then the language is understood to contain state variables as well). The language HL($@, \downarrow, E, \Diamond^-$), which we will also refer to as *the full hybrid language* (FHL), provides a natural upper bound on expressive power of hybrid languages: it is known to be expressively complete for first-order logic. In other words, every formula of the first-order correspondence language is equivalent to the standard translation of a FHL-formula.

So far, we have only introduced uni-modal HL($@, \downarrow$). This was only for convenience of exposition. It is straightforward to extend the above definitions to the multi-modal case. In fact, in the remainder of this paper, we will frequently make use of multi-modal formulas.

3 The Undecidability of HL($@, \downarrow$) Revisited

In this section, we revisit the negative result that is central to this paper: the undecidability of HL($@, \downarrow$) [9]. We present a new undecidability proof based on an encoding of the $\mathbb{N} \times \mathbb{N}$ tiling problem. It will help us identify the real source of the undecidability.

Let us first recall the $\mathbb{N} \times \mathbb{N}$ tiling problem. A tile type is a square, fixed in orientation, each side of which has a color. Formally, it can be identified with a 4-tuple of elements of some finite set of colors. To tile a space, we have to ensure that adjacent tiles have the same color on the matching sides. The $\mathbb{N} \times \mathbb{N}$ tiling problem is then: *given a finite set of tile types T, can the infinite grid $\mathbb{N} \times \mathbb{N}$ be tiled using only tiles of the types in T?* This problem is well known to be undecidable (see, e.g., [10]).

We will reduce this problem to the satisfiability problem for HL($@, \downarrow$) with three modalities: $\Diamond_1$ (to move one step up in the grid), $\Diamond_2$ (to move one step

to the right in the grid), and $\Diamond$ (to reach all the points of the grid), interpreted by the accessibility relations R_1, R_2 and R, respectively. Let T be a finite set of tiles, and for each tile $t \in T$ let $\mathrm{left}(t)$, $\mathrm{right}(t)$, $\mathrm{top}(t)$, and $\mathrm{bottom}(t)$ denote the four colors of t. We will now give a hybrid formula π_T that describes a tiling of $\mathbb{N} \times \mathbb{N}$ using the tile types in T. Note that the formula π_T given below is not the simplest possible encoding of the tiling problem. The reason is that the specific syntactic shape of π_T will be further exploited later on in the paper.

Spypoint $\alpha = s \wedge \Diamond s \wedge \Box\Diamond s \wedge \Box\Box_1 \downarrow x.(\Diamond(s \wedge \Diamond x)) \wedge \Box\Box_2 \downarrow x.(\Diamond(s \wedge \Diamond x))$, where s is a nominal. This formula says that the current world is named s, that the set of its R-successors is closed under R_1 and R_2, and that each R-successor of s has s as an R-successor.

Functionality $\beta = \bigwedge_{i=1,2} \big(\Box\Diamond_i \top \wedge \Box \downarrow x.\Box(s \rightarrow \Box(\Box_i x \vee \Box_i \neg x))\big)$. This formula, which is equivalent to $\bigwedge_{i=1,2} \big(\Box\Diamond_i \top \wedge \Box \downarrow x.\Box(s \rightarrow \Box(\Diamond_i x \rightarrow \Box_i x))\big)$, says that, within the submodel consisting of all R-successors of s, the relations R_1 and R_2 are in fact total functions.

Grid $\gamma = \Box \downarrow x.\Box(s \rightarrow \Box(\Box_1\Box_2 \neg x \vee \Box_2\Box_1 x))$. This formula, which, in the presence of functionality is equivalent to $\Box \downarrow x.\Box(s \rightarrow \Box(\Diamond_1\Diamond_2 x \rightarrow \Diamond_2\Diamond_1 x))$, expresses that R_1 and R_2 commute.

Tiling $\delta = \Box(\delta_1 \wedge \delta_2 \wedge \delta_3)$, where

$$\begin{aligned}
\delta_1 &= \bigvee_{t \in T}(p_t \wedge \bigwedge_{t' \in T; t \neq t'} \neg p_{t'}) \\
\delta_2 &= \bigwedge_{t \in T}(p_t \rightarrow \Box_2 \bigvee_{t' \in T; \mathrm{left}(t') = \mathrm{right}(t)} p_{t'}) \\
\delta_3 &= \bigwedge_{t \in T}(p_t \rightarrow \Box_1 \bigvee_{t' \in T; \mathrm{bottom}(t') = \mathrm{top}(t)} p_{t'})
\end{aligned}$$

Formula δ_1 states that exactly one tile is placed at each node of the grid, δ_2 says that horizontally adjacent tiles must match, and δ_3 says that vertically adjacent tiles must match. Hence, δ states that the grid is well-tiled.

It is easy to prove that T tiles $\mathbb{N} \times \mathbb{N}$ iff the hybrid formula $\pi_T = \alpha \wedge \beta \wedge \gamma \wedge \delta$ is satisfiable.

Notice that the formula π_T does not contain any @-operators, it does not nest the $\downarrow$ binder, and it uses only one state variable. Hence, the source of undecidability for hybrid logic is neither the @-operator, nor the nesting degree of $\downarrow$, nor the number of state variables used the formulas. Instead, as we will show in the next section, the source of undecidability is the $\Box\downarrow\Box$-pattern of β and γ (i.e., a $\Box$-operator scoping over a $\downarrow$ that in turn has scope over a $\Box$-operator). For formulas not containing this pattern, the satisfiability problem is decidable.

We conclude this section by briefly surveying undecidability proofs for hybrid logic with $\downarrow$ binder. The first undecidability proofs appear in [9, 11]. Both the proofs reduce an undecidable tiling problem into the satisfiability for hybrid logic with $\downarrow$ binder. The reduction of Goranko [11] uses a global modality, whereas Blackburn and Seligman [9] eliminate the use of a global modality by means of a spy point construction (cf. the formula α above). The encoding of [9] uses nested occurrences of the $\downarrow$ binder. Areces, Blackburn, and Marx [12] give another undecidability proof by a reduction of the undecidable global satisfaction problem

for K_{23} (the class of frames in which every state has at most 2 successors and at most 3 two-step successors). This proof has the advantage that it uses no nominals and no proposition letters. However, it does use nested occurrences of $\downarrow$. Finally, Marx [13] gives another proof of undecidability by a reduction from a tiling problem. The formulas used in this proof do not nest $\downarrow$ and contain only one state variable. Moreover, only one modality is used. However, the encoding is more involved than ours. Each of these proofs use formulas containing the $\Box\downarrow\Box$-pattern. The proof given above is reasonably simple, and it will help show the precise role of the $\Box\downarrow\Box$-pattern.

4 Syntactic Restrictions

In this section, we will show that the undecidability of $\mathrm{HL}(@,\downarrow)$ is caused by formulas containing the $\Box\downarrow\Box$-pattern. We show that without such formulas, the satisfiability problem is still decidable, even when the global modality and converse modalities are added to the language.

Consider the full hybrid language FHL. In what follows, it will be convenient to consider the universal operators $\Box$ and $\Box^-$, and the disjunction $\vee$, to be primitive operators (rather than shorthand notations). Moreover, we will restrict attention to sentences, i.e., formulas with no free state variables. This is not an essential limitation, since one can always replace free variables by nominals.

We say that a formula of FHL is in *negation normal form* (NNF) if the negation symbol appears only in front of atomic subformulas. Each hybrid formula is equivalent to a hybrid formula in NNF. For instance, $\neg\downarrow x.\,\Diamond(x\wedge\neg p)$ is equivalent to $\downarrow x.\,\Box(\neg x\vee p)$.

We call *universal operators* the modalities $\Box$, $\Box^-$ and A, and *existential operators* the modalities $\Diamond$, $\Diamond^-$ and E. We define a $\Box\downarrow$-formula (respectively, $\Diamond\downarrow$-formula) as a hybrid formula in NNF in which some occurrence of $\downarrow$ is in the scope of a universal (respectively, existential) operator. Moreover, we define a $\downarrow\Box$-formula (respectively, $\downarrow\Diamond$-formula) as a hybrid formula in NNF in which an occurrence of a universal (respectively, existential) operator is in the scope of a $\downarrow$. Similar definitions hold for different patterns. For example, $\Box\downarrow\Box$-formula is a formula in NNF containing a universal operator that contains in its scope a $\downarrow$ that contains in its scope a universal operator. A $\downarrow$-formula is simply a formula in NNF containing a $\downarrow$ binder. Given a pattern π, we define $\mathrm{FHL}\setminus\pi$ to be the fragment of FHL consisting of all formulas in NNF that are *not* of the form π. Notice that such fragments are not necessarily closed under negation.

Theorem 1. *There exists a polynomial satisfiability-preserving translation from* $\mathrm{FHL}\setminus\Box\downarrow$ *to* $\mathrm{HL}(@,\Diamond^-,E)$. *Moreover, the translation preserves satisfiability relative to any class of frames.*

Proof. It is convenient to introduce a new hybrid binder $\exists$. We add to the language formulas of the form $\exists x.\alpha$, where x is a state variable, interpreted as follows:

$$M,g,w\Vdash\exists x.\alpha \text{ iff } M,g^x_v,w\Vdash\alpha \text{ for some state } v$$

Notice that $\downarrow$ can be defined in terms of $\exists$ as follows: $\downarrow x.\alpha \equiv \exists x.(x \wedge \alpha)$.

Let us proceed with the proof. Let α_0 be a hybrid formula in FHL $\setminus \Box\downarrow$. We show how to polynomially translate α_0 into a formula α_3 in HL$(@, \Diamond^-, E)$ such that α_0 is satisfiable if, and only if, α_3 is satisfiable. The translation consists of three steps:

1. Let α_1 be obtained from α_0 be replacing each subformula of the form $\downarrow x.\varphi$ by $\exists x(x \wedge \varphi)$. Since no occurrence of the $\downarrow$ binder in α_0 is in the scope of a universal operator, the same holds for the occurrences of the $\exists$ binder in α_1;
2. rewrite α_1 into quantifier prefix form (i.e., where all occurrences of $\exists$ are in front of the formula), using the following equivalences: $\Diamond\exists x.\varphi \equiv \exists x.\Diamond\varphi$, $\Diamond^-\exists x.\varphi \equiv \exists x.\Diamond^-\varphi$, $E\,\exists x.\varphi \equiv \exists x.E\varphi$, $@_t\exists x.\varphi \equiv \exists x.@_t\varphi$, $\psi \wedge \exists x.\varphi \equiv \exists x.(\psi \wedge \varphi)$, $\psi \vee \exists x.\varphi \equiv \exists x.(\psi \vee \varphi)$. Note that renaming of variables might be necessary. Let α_2 be the resulting formula;
3. Let α_3 be obtained from α_2 by replacing each state variable by a fresh nominal and removing the corresponding existential quantifiers.

The resulting formula α_3 is in HL$(@, \Diamond^-, E)$, the length of α_3 is linear in the length of α_0, and α_0 and α_3 are easily seen to be equi-satisfiable. $\Box$

To illustrate the above proof, consider the formula $\downarrow x.\,\Diamond\downarrow y.\,@_x(\Diamond(y \wedge q) \wedge \Box(\Box\neg y \vee p))$, which does not contain the $\Box\downarrow$-pattern. It can be rewritten as follows:

$$\begin{array}{ll} \downarrow x.\,\Diamond\downarrow y.\,@_x(\Diamond(y \wedge q) \wedge \Box(\Box\neg y \vee p)) & \equiv \\ \exists x.\,(x \wedge \Diamond\exists y.\,(y \wedge @_x(\Diamond(y \wedge q) \wedge \Box(\Box\neg y \vee p)))) & \equiv \\ \exists x.\exists y.\,(x \wedge \Diamond(y \wedge @_x(\Diamond(y \wedge q) \wedge \Box(\Box\neg y \vee p)))) & \cong \\ i \wedge \Diamond(j \wedge @_i(\Diamond(j \wedge q) \wedge \Box(\Box\neg j \vee p))) & \end{array}$$

Corollary 1. *The satisfiability problem for* FHL $\setminus \Box\downarrow$ *is* ExpTime*-complete.*

Proof. The lower bound follows from the fact that FHL $\setminus \Box\downarrow$ embeds the basic modal language with global modality, which is known to have an ExpTime-complete satisfiability problem [14]. The upper bound follows from Theorem 1 since satisfiability of HL$(@, \Diamond^-, E)$-formulas can be decided in ExpTime [15].

We now prove the mirror image of Theorem 1: satisfiability for FHL $\setminus \downarrow\Box$ is decidable. The technique we use is similar to the one used by Marx [13]: we embed FHL$\setminus\downarrow\Box$ into the $\forall$-guarded fragment. The $\forall$-guarded fragment of first-order logic consists of all formulas constructed from atomic formulas and their negations using conjunction, disjunction, existential quantification, and guarded universal quantification. Hence only the universal quantification is constrained. The satisfiability problem for $\forall$-guarded first-order formulas is 2ExpTime-complete. It is ExpTime-complete when there is a uniform bound on the width of the formula. For more details, cf. [16].

Theorem 2. *The satisfiability problem for* FHL$\setminus\downarrow\Box$ *is in* 2ExpTime*. The satisfiability problem for* FHL$\setminus\downarrow\Box$*-formulas of bounded width is* ExpTime*-complete.*

Proof. Let α be any FHL $\setminus \downarrow\Box$-sentence. We will show by induction on α that $\mathrm{ST}_x(\alpha)$ is $\forall$-guarded. Since $\mathrm{ST}_x(\alpha)$ can be obtained from α in polynomial time, this proved that the satisfiability problem for FHL $\setminus \downarrow\Box$ is in 2ExpTime.

To smoothen the induction, we will prove the result for any subformula α of a FHL $\setminus \downarrow\Box$-sentence. If α is a (negated) atomic formula, then $\mathrm{ST}_x(\alpha)$ is quantifier-free, hence $\forall$-guarded. If α is of the form $\alpha_1 \wedge \alpha_2$ or $\alpha_1 \vee \alpha_2$, then by the induction hypothesis, $\mathrm{ST}_x(\alpha)$ is the conjunction (respectively, disjunction) of two $\forall$-guarded formulas, and hence is $\forall$-guarded.

Next, suppose α is of the form $X\alpha_1$, where X is an existential operator or an @-operator. By the induction hypothesis, $\mathrm{ST}_y(\alpha_1)$ is $\forall$-guarded. Inspection of the relevant clauses of the Standard Translation shows that $\mathrm{ST}_x(\alpha)$ is also $\forall$-guarded.

Next, suppose α is of the form $X\alpha_1$, where X is a universal operator. Again, by induction hypothesis, $\mathrm{ST}_y(\alpha_1)$ is $\forall$-guarded. Moreover, by assumption α is a subformula of a FHL $\setminus \downarrow\Box$-sentence. It follows α_1 cannot contain any free state variables (for, these would have to be bound higher up). It follows that $\mathrm{ST}_y(\alpha_1)$ contains no free variables besides (possibly) y. Inspection of the relevant clauses of the Standard Translation shows that this variable y is appropriately guarded in $\mathrm{ST}_x(\alpha)$, and hence $\mathrm{ST}_x(\alpha)$ is $\forall$-guarded.

Finally, suppose α is of the form $\downarrow z.\alpha_1$. Then, $\mathrm{ST}_x(\alpha) = \exists z.(z = x \wedge \mathrm{ST}_x(\alpha_1))$. By induction hypothesis, $\mathrm{ST}_x(\alpha_1)$ is $\forall$-guarded. It follows that $\mathrm{ST}_x(\alpha)$ is also $\forall$-guarded.

It is easy to see that, if a hybrid formula α has width w, then the width of $\mathrm{ST}_x(\alpha)$ is at most $w + 2$. Hence, a bound on the width of the FHL $\setminus \downarrow\Box$-formula implies a bound on the width of its $\forall$-guarded standard translation. Since the satisfiability problem for $\forall$-guarded formulas of bounded width is ExpTime-complete, this gives us an ExpTime upper bound. The lower bound follows from the ExpTime-hardness of the basic modal logic extended with the global modality [14]. □

Satisfiability for FHL $\setminus \downarrow\Box$ is ExpTime-hard, since satisfiability for modal logic with the global modality is already ExpTime-hard [14]. We don't know the exact complexity of FHL $\setminus \downarrow\Box$, but we conjecture that it is ExpTime-complete.

By combining the techniques used to prove Theorems 1 and 2, we have the main result of this section:

Theorem 3. *The satisfiability problem for* FHL $\setminus \Box\downarrow\Box$ *is in* 2ExpTime. *The satisfiability problem for* FHL $\setminus \Box\downarrow\Box$*-formulas of bounded width is* ExpTime*-complete.*

Proof. Let $\alpha \in$ FHL $\setminus \Box\downarrow\Box$. If $\alpha \in$ FHL $\setminus \downarrow\Box$, then the satisfiability of α can be decided in 2ExpTime by Theorem 2. Suppose therefore that $\alpha \notin$ FHL $\setminus \downarrow\Box$. Let β be a minimal $\downarrow\Box$-subformula of α. Since $\alpha \in$ FHL $\setminus \Box\downarrow\Box$, β cannot be in the scope of a universal operator in α. It follows that this occurrence of $\downarrow$ can be removed as in the proof of Theorem 1. Repeating this step for each minimal $\downarrow\Box$-subformula of α, we obtain a formula $\beta \in$ FHL $\setminus \downarrow\Box$ that is satisfiable iff α is satisfiable. By Theorem 2, satisfiability of β can be checked in 2ExpTime. The

ExpTime-completeness in the case of bounded width follows from the bounded width case of Theorem 2. □

To illustrate the above proof, consider the formula $\alpha = \Diamond\downarrow x.\Box\downarrow y.@_y\Diamond x$. It contains both the $\downarrow\Box$- and the $\Box\downarrow$-pattern, hence neither Theorem 1 nor Theorem 2 can be applied. However, α does not contain the $\Box\downarrow\Box$-pattern, hence Theorem 3 can be invoked. There exists only one minimal $\downarrow\Box$-subformula of α, that is $\beta = \downarrow x.\Box\downarrow y.@_y\Diamond x$. The outermost occurrence of $\downarrow$ in β is not in the scope of any universal operator in α, hence it can be removed as done in Theorem 1. The resulting equi-satisfiable formula is $\alpha' = \Diamond(i \wedge \Box\downarrow y.@_y\Diamond i)$, which does not contain the $\downarrow\Box$-pattern anymore. Hence Theorem 2 can be applied to it.

Since the negation of an FHL $\setminus \Diamond\downarrow\Diamond$-formula is equivalent to an FHL $\setminus \Box\downarrow\Box$-formula, we have as a corollary the following dual result.

Corollary 2. *The validity problem for* FHL$\setminus\Diamond\downarrow\Diamond$ *is in 2*ExpTime*. The validity problem for* FHL $\setminus \Diamond\downarrow\Diamond$*-formulas of bounded width is* ExpTime*-complete.*

In particular, if a hybrid formula ϕ contains neither the $\Box\downarrow\Box$ pattern nor the $\Diamond\downarrow\Diamond$ pattern, then both satisfiability and validity of ϕ are decidable.

5 Semantic Restrictions

In this section, we restrict attention to uni-modal models of bounded width, i.e., models with only one binary relation R, in which each node is R-related only to a restricted number of points. More precisely, for any cardinal κ, let K_κ be the class of uni-modal models in which for every node d there are strictly less than κ nodes e such that $(d, e) \in R$. In particular, K_2 is the class of models in which every points has at most one R-successor, and K_ω is the class of models in which every node has only finitely many R-successors. We will refer to elements of K_κ as κ-models for short. In what follows we will consider the satisfiability problem of HL$(@, \downarrow)$ and of the first-order correspondence language on κ-models, for particular κ. Our results are summarized in Table 1. All results generalize to the case with multiple modalities, except for the decidability of the first-order correspondence language on K_2.

The terminology and results used in this section can be found in [17] and [10], or in other texts on computational complexity. In particular, we follow the usual terminology from recursion theory: the language of second-order arithmetic is the second-order language with constants 0, 1, function symbols + and $\times$, and equality. Formulas of second-order arithmetic are interpreted over the natural numbers. A Σ^1_1 formula of second order arithmetic is a formula of the form $\exists R_1 \ldots R_n.\phi$ where ϕ contains no second-order quantifiers. A set A of natural numbers is said to be in Σ^1_1 if it is defined by a Σ^1_1 formula that has one free first-order variable and no free second-order variables. A set A of natural numbers is Σ^1_1-hard if for every B in Σ^1_1 there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for all $n \in \mathbb{N}$, $n \in B$ iff $f(n) \in A$. A set of natural numbers is Σ^1_1-complete if it is both in Σ^1_1 and Σ^1_1-hard. It is well known that Σ^1_1-hard sets are not recursively

Table 1. Complexity of the satisfiability problem on κ-models.

	HL($@, \downarrow$)	first-order correspondence language
$\kappa = 1$	NP-complete	NExpTime-complete
$\kappa = 2$	NP-complete	Decidable, not elementary recursive
$3 \leq \kappa < \omega$	NExpTime-complete	Π^0_1-complete (co-r.e., not decidable)
$\kappa = \omega$	Σ^0_1-complete (r.e., not decidable)	Σ^1_1-complete (highly undecidable)
$\kappa > \omega$	Π^0_1-complete (co-r.e., not decidable)	Π^0_1-complete (co-r.e., not decidable)

enumerable. When one speaks of an arbitrary decision problem as being in Σ^1_1 or Σ^1_1-hard, it is implicitly understood that the instances of the decision problem are coded into natural numbers (under some computable encoding).

Following [17], we call a decidable problem *elementary recursive* if the time complexity can be bounded by a constant number of iterations of the exponential function.

Theorem 4. *The satisfiability problem of* HL($@, \downarrow$) *on* K_κ *is:*

1. NP*-complete, for* $\kappa = 1, 2$.
2. NExpTime*-complete, for* $3 \leq \kappa < \omega$.
3. *Recursively enumerable but not decidable, for* $\kappa = \omega$.
4. *Co-recursively enumerable but not decidable, for* $\kappa > \omega$.

Proof. Point 1. The lower bound follows from the NP-hardness of propositional satisfiability. The upper bound is proved by establishing the polynomial size model property.

For $\kappa = 1, 2$, every κ-satisfiable HL($@, \downarrow$)-formula is satisfiable in a κ-model with at most $O(|\phi|^2)$ nodes. For, suppose $M, g, w \Vdash \phi$ for some κ-model $M = (W, R, V)$ and assignment g. Let $W' \subseteq W$ consist of all worlds that are reachable from w or from a world named by one of the nominals occurring in ϕ in at most $md(\phi)$ steps, where $md(\phi)$ is the modal depth of ϕ. Let M' be the submodel of M with domain W'. Clearly, M' is a κ-model and M' satisfies the cardinality requirements and a straightforward induction argument shows that $M', g, w \Vdash \phi$.

This leads to a non-deterministic polynomial time algorithm for testing satisfiability of an HL($@, \downarrow$)-formula ϕ on κ-models, for $\kappa = 1, 2$. The algorithm first non-deterministically chooses a candidate model (M, g, w) of size $O(|\phi|^2)$, and then it tests whether $M, g, w \Vdash \phi$ and $M \in \mathsf{K}_\kappa$. The latter tests can be performed in polynomial time using a top down model checking algorithm (cf. Theorem 6).

Point 2 (Upper bound). For $3 \leq \kappa < \omega$, every formula satisfiable on a κ-model is satisfiable on a κ-model with at most $O(|\phi| \cdot \kappa^{md(\phi)})$ nodes. For, suppose $M, g, w \Vdash \phi$ for some κ-model $M = (W, R, V)$ and assignment g. Let $W' \subseteq W$ consist of all worlds that are reachable from w or from a world named by one of the nominals occurring in ϕ in at most $md(\phi)$ steps. Let M' be the submodel of M with domain W'. Note that the cardinality of M' is $O(|\phi| \cdot \kappa^{|\phi|})$, and M' is still a κ-model. Furthermore, a straightforward induction argument shows that $M', g, w \Vdash \phi$.

This leads to a non-deterministic ExpTime algorithm for testing satisfiability of an HL($@, \downarrow$)-formula ϕ on κ-models. The algorithm first non-deterministically chooses a candidate model (M, g, w) of size $O(|\phi| \cdot \kappa^{|\phi|})$, and then tests whether $M, g, w \Vdash \phi$. The latter test can be performed in time $O(|M|^{|\phi|})$ [8], which is $O((|\phi| \cdot \kappa^{|\phi|})^{|\phi|}) = O(|\phi|^{|\phi|} \cdot \kappa^{(|\phi|^2)})$.

Point 2 (Lower bound). Consider monadic first-order formulas without equality, i.e., first-order formulas containing unary predicates only, without equality. Any such satisfiable formula ϕ of length n has a model with at most 2^n nodes, and the satisfiability problem for such formulas is NExpTime-complete [17, Section 6.2.1]. We will reduce this problem to the satisfiability problem for HL($@, \downarrow$)-formulas on κ-models (for $3 \leq \kappa < \omega$), thus showing that the latter problem is NExpTime-hard.

Fix a nominal i, and for any monadic first-order formula ϕ without equality, define ϕ^+ inductively, such that $(x = y)^+ = @_x y$, $(Px)^+ = @_x p$, $(\cdot)^+$ commutes with the Boolean connectives and $(\exists x.\psi)^+ = @_i \Diamond^{|\phi|} \downarrow x.\psi$. In words, ϕ^+ states that ϕ holds in the submodel consisting of all points reachable from the point named i in exactly $|\phi|$ many steps. In general, there can be up to $(\kappa - 1)^{|\phi|}$ many points reachable from the point named i in exactly $|\phi|$ many steps (in particular, this will be the case if the submodel generated by i is a $(\kappa - 1)$-ary tree). Thus, ϕ is satisfiable iff ϕ is satisfiable in a model with at most $2^{|\phi|}$ nodes iff ϕ^+ is satisfiable in a κ-model, for $\kappa \geq 3$.

Point 3. We will provide polynomial reductions between this problem and the finite satisfiability problem for first-order logic. The satisfiability problem for first-order logic on finite models is Σ_1^0-complete, even in the case with only a single, binary relation [17, Section 3.2].

Trivially, if an HL($@, \downarrow$)-formula is satisfiable in a finite model, it is satisfiable in an ω-model. Conversely, if an HL($@, \downarrow$)-formula is satisfiable in an ω-model then is satisfiable in a finite model, since the modal depth of the formula provides a bound on the depth of the model. Hence, the satisfiability problem of HL($@, \downarrow$) on ω-models reduces (by the Standard Translation) to the satisfiability problem for first-order logic on finite models.

Conversely, the finite satisfiability problem for first-order logic can be reduced to satisfiability of HL($@, \downarrow$) on ω-models. Fix a nominal i, and for any first-order formula ϕ, define ϕ^+ inductively, such that $(x = y)^+ = @_x y$, $(Rxy)^+ = @_x \Diamond y$, $(\cdot)^+$ commutes with the Boolean connectives and $(\exists x.\psi)^+ = @_i \Diamond \downarrow x.\psi^+$. In words, ϕ^+ states that ϕ holds in the submodel consisting of the successors of the point named i. It follows that ϕ is satisfiable in a finite model iff the HL($@, \downarrow$)-formula ϕ^+ is satisfiable on an finitely branching ω-model.

Point 4. By the Löwenheim-Skolem theorem, a first-order formula is satisfiable if and only if it is satisfiable on a finite or countably infinite model. Since HL($@, \downarrow$) is a fragment of first-order logic, the Löwenheim-Skolem theorem also applies to HL($@, \downarrow$)-formulas. It follows that the satisfiability problem for HL($@, \downarrow$) on countably branching models coincides with the general satisfiability problem of HL($@, \downarrow$), which is in Π_1^0 by the Standard Translation and Π_1^0-hard by the tiling argument from Section 3. □

As the following theorem shows, the first-order correspondence language performs much worse.

Theorem 5. *The satisfiability problem of first-order sentences of the correspondence language on* K_κ *is:*

1. NExpTime *complete, for* $\kappa = 1$.
2. *decidable but not elementary recursive, for* $\kappa = 2$.
3. *Co-recursively enumerable but not decidable, for* $3 \leq \kappa < \omega$.
4. Σ^1_1*-hard, and hence neither recursively enumerable nor co-recursively enumerable, for* $\kappa = \omega$.
5. *Co-recursively enumerable but not decidable, for* $\kappa > \omega$.

Proof. We prove here only the decidable cases (points 1 and 2). The reader is referred to the full version of this paper [18] for a full proof of the theorem.

Point 1. This case coincides with the satisfiability problem for monadic first-order logic (on 1-models, every formula of the form Rst is equivalent to $\bot$), which is known to be NExpTime complete [17].

Point 2. Consider the satisfiability problem for first-order logic with one unary function symbol, an arbitrary number of unary relation symbols and equality ("the Rabin class"). This problem is decidable, but not elementary recursive [17]. We will provide reductions between this problem and the satisfiability problem for first-order logic on 2-models.

Let ϕ be any first-order formula containing one unary function symbol f and any number of unary relation symbols and equality. Let R be a binary relation symbol, and let ϕ_R be obtained from ϕ by repeatedly applying the rewrite rules

- replace atomic formulas of the form $Pf(t)$ by $\exists x.(Rtx \wedge Px)$
- replace atomic formulas of the form $f(s) = t$ or $t = f(s)$ by $\exists x.(Rsx \wedge x = t)$

until the function symbol f does not occur in the formula anymore (in case of nested function symbols, the above rules might need to be applied several times). It is not hard to see that ϕ is satisfiable iff $\phi_R \wedge \forall x \exists y.Rxy$ is satisfiable on a 2-model.

Let ϕ be any first-order formula with one binary relation symbol R and any number of unary relation symbols. Let f be a unary function symbol and let P be a new unary relation, and let ϕ_f be the result of replacing all subformulas of ϕ of the form Rst by $Ps \wedge (t = fs)$. Intuitively, the unary predicate P represents the existence of a successor, and the unary function f encodes the successor of a node, if it exists. One can easily see that ϕ is satisfiable on a 2-model iff ϕ_f is satisfiable (simply let R denote the graph of f, or viceversa).

It follows that the satisfiability problem of first-order logic on 2-models is decidable but not elementary recursive. □

6 Model Checking

So far, we only studied the satisfiability and the validity problems. It is natural to ask how our syntactic and semantic restrictions affect the complexity of the model checking problem.

Given a hybrid model M, an assignment g, a state w, and a hybrid formula α, the *model checking problem* is to check whether $M, g, w \Vdash \alpha$. We will restrict ourselves to hybrid sentences. This is not a limitation, since one can always replace the free variables by fresh nominals, expanding the model accordingly.

In [8], the authors give a polynomial time model checker for $\mathrm{HL}(@, \Diamond^-, E)$. Moreover, they prove that the model checking problem for $\mathrm{HL}(@, \downarrow)$ is PSPACE-complete (as it is for full first-order logic), even for formulas without nominals, @-operators and proposition letters.

Theorem 6. *The model checking problem for* $\mathrm{HL}(@, \downarrow)$ *on* κ*-models can be solved in polynomial time for* $\kappa \leq 2$*, and is* PSPACE*-complete for* $\kappa \geq 3$*.*

Proof. The first part of the theorem can be proved using a straightforward top-down model checking algorithm. Since each state in the model has at most one successor, the algorithm takes time linear in the length of the input formula. As for the second part, the proof of PSPACE-hardness of model checking for $\mathrm{HL}(@, \downarrow)$ given in [8] uses a model with out-degree 2. It follows that the model checking problem for $\mathrm{HL}(@, \downarrow)$ on κ-models, with $\kappa \geq 3$, is PSPACE-complete. □

For $\mathrm{HL}(@, E, \downarrow)$ and first-order logic, on the other hand, the model checking problem is PSPACE-complete even on 1-models [8].

In the following, we investigate how restrictions on the *syntax* of hybrid formulas affect the complexity of model checking. Our first result is that, if formulas containing the $\downarrow\Box\downarrow$ pattern are excluded, then the model checking problem drops from PSPACE to NP.

Theorem 7. *The model checking problem for* $\mathrm{FHL} \setminus \downarrow\Box\downarrow$ *is* NP*-complete.*

Proof. To prove NP-hardness, we embed the satisfiability problem for propositional formulas (SAT) into the model checking problem for $\mathrm{FHL} \setminus \downarrow\Box\downarrow$. Let $\phi(p_1, \ldots, p_n)$ be any propositional formula, and let $M = (W, R, V)$, where $W = \{0, 1\}$ and $R = W \times W$ (the valuation V is irrelevant). For each p_k occurring in ϕ, pick a corresponding state variable x_k. Furthermore, let y be a state variable distinct from all $x_1, \ldots, x_n$. Let ϕ' be obtained from ϕ by replacing each occurrence of a proposition letter p_k by $\Diamond(x_k \wedge y)$. Intuitively, the two states of M represent truth and falsity, and among these two states the variable y denotes the truth state. It is easily seen that the propositional formula ϕ is satisfiable iff $\Diamond\downarrow y \Diamond\downarrow x_1 \Diamond\downarrow x_2 \ldots \Diamond\downarrow x_n.\phi'$ is true in M (at any of the nodes $0, 1$). The latter formula contains no universal operators, and hence belongs to $\mathrm{FHL} \setminus \downarrow\Box\downarrow$.

To show that the problem is in NP, we give a nondeterministic algorithm that solves the model checking problem in polynomial time. Let α be an $\mathrm{FHL} \setminus \downarrow\Box\downarrow$ sentence, $M = (W, R, V)$ be a model, $v \in W$ and g be an assignment. Replace each subformula of α of the form $\downarrow x.\varphi$ by $\exists x.(x \wedge \varphi)$, and apply the equivalences given in the proof of Theorem 1 in order to move the existential quantifiers out of the scope of as many connectives as possible. The resulting sentence α' is equivalent to α and has the following properties:

1. α' is built up from literals (i.e., formulas of the form $(\neg)p$, $(\neg)i$ or $(\neg)x$) using conjunction, disjunction, existential operators $(\Diamond, \Diamond^-, E)$, universal operators $(\Box, \Box^-, A)$, and existential quantifiers.
2. All existential quantifiers in α' either immediately follow a universal operator (e.g., as in $\Box\exists x_1 \dots x_n\gamma$) or occur at the start of the formula.
3. For all subformulas of α' of the form $X\exists x_1 \dots x_n\gamma$, with X a universal operator, γ contains no free variables besides $x_1, \dots, x_n$.

List all subformulas of α' of the form $X\beta$, with X a universal operator and $\beta = \exists x_1 \dots \exists x_m.\gamma(x_1 \dots x_m)$, in order of increasing length. For each such β do the following: create a new proposition symbol p_β and replace β by p_β in α'. For each state $w \in W$, check whether $M, g, w \Vdash \beta$, and, if the answer is positive, then insert the state w in $V(p_\beta)$.

The nondeterminism is hidden in the test $M, g, w \Vdash \beta$. Indeed, to check whether $M, g, w \Vdash \exists x_1 \dots \exists x_m.\gamma(x_1 \dots x_m)$, the algorithm guesses an assignment $w_1, \dots, w_m$ for the variables $x_1, \dots, x_m$, respectively, and then it checks whether $M, g^{x_1,\dots,x_m}_{w_1,\dots,w_m}, w \Vdash \gamma(x_1 \dots x_m)$. Since γ does not contain any existential quantifiers (the subformulas were processed in order of increasing length), it belongs to $\mathrm{HL}(@, \Diamond^-, E)$ and hence the model checking can be performed in polynomial time.

The resulting formula is in $\mathrm{HL}(@, \Diamond^-, E)$ and thus it can be model checked in polynomial time. □

Notice that the NP-hardness holds even for formulas without proposition letters, nominals and @-operators. A typical example of a formula to which Theorem 7 does not apply is $\downarrow x.\Box\Box\downarrow y.@_x\Diamond y$, which expresses a local form of transitivity.

In Section 4, we saw that $\mathrm{FHL} \setminus \Box\downarrow\Box$ has a decidable satisfiability problem. We leave it as an open question whether the model checking complexity of that fragment is also below PSPACE (since the SAT problem can be embedded into the model checking problem for $\mathrm{FHL} \setminus \Box\downarrow\Box$ as done in the proof of Theorem 7, the problem is at least NP-hard). Conversely, the fragment $\mathrm{FHL} \setminus \downarrow\Box\downarrow$, for which we have just proved that the model checking problem is NP-complete, has an undecidable satisfiability problem: it suffices to note that the encoding of the tiling problem given in Section 3 does not make use of $\downarrow\Box\downarrow$-formulas.

We conclude this section with a hierarchy of fragments of the full hybrid language with $\downarrow$ binder that admits polynomial time model checking. If a hybrid formula α has width w, then $\mathrm{ST}_x(\alpha)$ has width at most $w+2$. Hence, a bound on the width of the hybrid formulas implies a bound on the width of the standard translations. Moreover, model checking for first-order formulas using a bounded number of variables can be performed in polynomial time [19]. It is known that first-order formulas of a bounded width can be rewritten using a bounded number of variables (cf. [20] for an explicit proof). Thus, we obtain the following.

Theorem 8. *The model checking problem for formulas of the full hybrid language of bounded width can be solved in polynomial time.*

7 Conclusion

In this paper, we described two ways to tame the power of hybrid logic with binders. These are: (i) restricting the syntax by excluding formulas containing the pattern $\Box\downarrow\Box$, and (ii) restricting the class of models by assuming a bound on the branching degree of the models. Furthermore, we showed that similar restrictions can be used to lower the complexity of the model checking task.

Our decidability result for FHL $\setminus \Box\downarrow\Box$ may be seen from a more general perspective: one could consider any sequence $\pi \subseteq \{\Box, \Diamond, \downarrow, @\}^*$, where $\Box$ stands for "a sequence of universal modalities", and $\Diamond$ stands for "a sequence of existential modalities", and ask whether the satisfiability problem for FHL $\setminus \pi$ is decidable. In particular, one could ask if there is such a sequence π that contains $\Box\downarrow\Box$ as a subsequence and such that the satisfiability problem for FHL $\setminus \pi$ is still decidable. Our undecidability proof in Section 3 (and more in particular the shape of the formulas β and γ used there) shows that the answer is negative, and hence Theorem 3 is tight.

Some results in this paper show that, under certain natural conditions, the language HL$(@, \downarrow)$ behaves better than the first-order correspondence language, computationally speaking. Incidentally, the full hybrid language FHL has the same expressive power as the first-order correspondence language, as was shown in [21] by means of a translation HT mapping formulas of the first-order correspondence to FHL-formulas. The most interesting clause of this translation says $\mathrm{HT}(\exists x.\phi) = E\downarrow x.\mathrm{HT}(\phi)$. It shows that, in some sense, the first-order quantifier $\exists x$ consist of two parts, namely the *picking a state of the model* part, which is captured by the global modality, and the *variable binding* part, which is captured by the $\downarrow$. The syntax of HL$(@, E, \downarrow, \Diamond^-)$ allows us to distinguish these two parts. One could say that our results identify computationally tractable fragments of first-order logic that can only be distinguished once these two parts of the quantifiers are split. In this sense, our paper can be seen as a fine study of the structure of first-order quantifiers.

Finally, the outcomes of our investigation show once more that, from a computational point of view, the satisfiability problem and the model checking problem for a logic are sensitive to different *sources* of complexity. Restricting the model width makes satisfiability easier, but it does not lower the complexity of model checking. On the other hand, restricting the formula width makes model checking more tractable, but it does not affect the undecidability of satisfiability.

Acknowledgements

This paper has benefited from discussions with Maarten Marx, and from the comments of the anonymous reviewers. The authors were supported by NWO grants 612.069.006 and 612.000.207, respectively.

References

1. Andréka, H., van Benthem, J., Németi, I.: Modal logics and bounded fragments of predicate logic. Journal of Philosophical Logic **27** (1998) 217–274
2. Grädel, E.: On the restraining power of guards. Journal of Symbolic Logic **64** (1999) 1719–1742
3. Mortimer, M.: On languages with two variables. Zeitschrift für mathematische Logik und Grundlagen der Mathematik **21** (1975) 135–140
4. Grädel, E., Otto, M.: On logics with two variables. Theoretical computer science **224** (1999) 73–113
5. Areces, C., Blackburn, P., Marx, M.: Hybrid logics: Characterization, interpolation, and complexity. Journal of Symbolic Logic **66** (2001) 977–1010
6. ten Cate, B.: Interpolation for extended modal languages. Journal of Symbolic Logic **70** (2005) 223–234
7. ten Cate, B.: Model theory for extended modal languages. PhD thesis, University of Amsterdam (2005) ILLC Dissertation Series DS-2005-01.
8. Franceschet, M., de Rijke, M.: Model checking for hybrid logics (with an application to semistructured data). Journal of Applied Logics (2005) To appear.
9. Blackburn, P., Seligman, J.: Hybrid languages. Journal of Logic, Language and Information **4** (1995) 251–272
10. Harel, D.: Recurring dominoes: making the highly undecidable highly understandable. Annals of Discrete Mathematics **24** (1985) 51–72
11. Goranko, V.: Hierarchies of modal and temporal logics with reference pointers. Journal of Logic, Language, and Information **5** (1996) 1–24
12. Areces, C., Blackburn, P., Marx, M.: A road-map on complexity for hybrid logics. In Flum, J., Guez-Artalejo, M.R., eds.: Proceedings of the 8th Annual Conference of the EACSL, Madrid (1999)
13. Marx, M.: Narcissists, stepmothers and spies. In: Proceedings of the International Workshop on Description Logics. (2002)
14. Fisher, M., Ladner, R.: Propositional dynamic logic of regular programs. Journal of Computer and System Sciences **18** (1979) 194–211
15. Areces, C., Blackburn, P., Marx, M.: The computational complexity of hybrid temporal logics. Logic Journal of the IGPL **8** (2000) 653–679
16. ten Cate, B., Franceschet, M.: Guarded fragments with constants. Journal of Logic, Language, and Information (2005) To appear.
17. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Springer, Berlin (1997)
18. ten Cate, B., Franceschet, M.: On the complexity of hybrid logics with binders. Technical Report PP-2005-02, ILLC, University of Amsterdam (2005)
19. Vardi, M.Y.: On the complexity of bounded-variable queries. In: Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. (1995) 266–276
20. ten Cate, B., Franceschet, M.: Guarded fragments with constants. Technical Report PP-2004-32, ILLC, University of Amsterdam (2004)
21. Blackburn, P.: Representation, reasoning, and relational structures: A hybrid logic manifesto. Logic Journal of the IGPL **8** (2000) 339–365