Complexity of Hybrid Logics over Transitive Frames

Martin Mundhenk[1], Thomas Schneider[1], Thomas Schwentick[2], and Volker Weber[2]

[1] Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany
{mundhenk, schneider}@cs.uni-jena.de

[2] Fachbereich Informatik, Universität Dortmund, Germany
{thomas.schwentick, volker.weber}@udo.edu

Abstract. This paper examines the complexity of hybrid logics over transitive frames and transitive trees. We show that satisfiability over transitive frames for the hybrid language extended with $\downarrow$ is NEXP-complete. This is in contrast to undecidability of satisfiability over arbitrary frames for this language [2]. We also show that adding @ or the past modality leads to undecidability over transitive frames, but not over transitive trees, where we show the richest language to be nonelementarily decidable. Moreover, we establish 2EXP and EXP upper bounds for satisfiability over transitive frames and transitive trees, respectively, for the hybrid $\mathsf{Until}/\mathsf{Since}$ language. An EXP lower bound is shown to hold for the modal Until language over both frame classes.

1 Introduction

We examine the computational complexity of satisfiability for hybrid logics over transitive frames. This is an important frame class, because transitivity is the minimal requirement in many temporal applications, for example temporal verification. Modal, hybrid, and first-order logic over transitive models have been studied recently in [3, 14, 30, 18, 19, 17, 11]. Although the complexity of hybrid (tense) logics has been extensively examined [7, 15, 2, 3, 13], there are highly expressive hybrid languages for whose satisfiability problems only results over arbitrary, but not over restricted, temporally relevant, frame classes have been known.

Hybrid Languages are extensions of the language of modal logic that allow for naming and accessing states of a model explicitly. This renders hybrid logic an adequate representation formalism for many applications where the basic modal and/or temporal language does not suffice. Moreover, reasoning systems are easier to devise for hybrid than for modal logic.

Hybrid logic, as well as the foundations of temporal logic, goes back to Arthur Prior [25]. Since then, many — more or less powerful — languages have been studied. Here we briefly introduce the extensions that shall concern us in this paper.

Nominals are special atomic formulae that name states of models. They allow, for instance, for an axiom expressing irreflexivity, which cannot be captured by modal formulae: $i \to \neg\Diamond i$.

The at operator @ can be used to directly jump to states named by nominals, independently of the accessibility relation. Hence, the above formula could also be written as $@_i\neg\Diamond i$.

With the help of *the downarrow operator* $\downarrow$, it is possible to bind variables to states. Whenever $\downarrow x$ is encountered during the evaluation of a formula, the variable x is bound to the actual state s. All occurrences of x in the scope of this $\downarrow$ are treated like nominals naming s. As an example, the formula $\downarrow x.\neg\Diamond\Diamond x$ reads as: Name the actual state x and make sure that it is not possible to reach x in two steps. This is an axiom for asymmetry, another property not expressible in modal logic.

Combined with the @ operator, $\downarrow$ leads to a very powerful language that can formulate many desirable properties and goes far beyond the scope of the simple nominal language. To give a more impressive example, we consider the Until operator. The formula $\mathsf{U}(\varphi, \psi)$ reads as "there is a point in the future at which φ holds, and at all points between now and this point, ψ holds". What the basic modal language is not able to express, can be achieved by the hybrid $\downarrow$-@ language.

$$\mathsf{U}(\varphi, \psi) \equiv \downarrow x.\Diamond\downarrow y.\varphi \wedge @_x\Box(\Diamond y \to \psi).$$

Besides more advanced temporal concepts such as "until" and "since", hybrid temporal languages can express other desirable temporal notions such as "now", "yesterday", "today", or "tomorrow". Moreover, with hybrid logic one can capture many temporally relevant frame properties (besides the above mentioned, antisymmetry, trichotomy, directedness, ...). For this reason, hybrid temporal languages are of great interest where basic temporal logic reaches its limits [9, 5, 4, 13].

Why Transitivity? Hybrid logic is interpreted over Kripke frames and models, as is modal logic. A frame consists of a set of states (points in time) and an accessibility relation (where xRy says that y is in the future of x). In most temporal applications one requires this relation to be transitive. We concentrate on transitive frames because transitivity is a property that the future relations of many different temporal applications have in common, even if they differ in other properties such as tree-likeness, trichotomy, irreflexivity, or asymmetry.

But there are other reasons why this frame class is of interest, particularly in connection with computational complexity. In the special case of linear frames, nominals and @ can be simulated using the conventional modal operator and its converse. They do not add expressive power to the language in this case. The $\downarrow$ operator is useless even on transitive trees, a representation of branching time. Though the class of transitive frames (and transitive trees for $\downarrow$-free hybrid languages, respectively) is a restricted frame class, it is general enough to separate hybrid from modal languages in terms of expressive power.

Yet another reason for considering precisely transitive frames will become clear in the next paragraph.

Complexity of Hybrid Logics. We use the complexity classes NP, PSPACE, EXP, NEXP, nEXP, $n \geq 2$, and coRE as known from [24]. A problem is nonelementarily decidable if it is decidable and not contained in any nEXP.

It goes without saying that reasoning tasks for richer logics require more resources than those for simpler languages such as the basic modal language. We focus on one reasoning task, namely satisfiability. The modal and temporal satisfiability problems over arbitrary as well as transitive frames are PSPACE-complete [22, 29]. If the "somewhere" modality E is added, satisfiability becomes EXP-complete over arbitrary frames [28]. For many, more restricted, frame classes, modal and temporal satisfiability is NP-complete [22, 23, 27]. In contrast, the known part of the complexity spectrum of hybrid satisfiability reaches up to undecidability.

Many complexity results for hybrid languages have been established in [2, 3]. It was shown in [2] that the hybrid language with nominals and @ has a PSPACE-complete satisfiability problem and that satisfiability for the hybrid tense language is EXP-complete, even if @ or E are added. It was proven in [3] that these problems have the same complexity (or drop to PSPACE-complete or NP-complete, respectively) if the class of frames is restricted to transitive frames (or transitive trees, or linear frames, respectively).

Moreover, the authors of [3] established EXP-completeness of satisfiability for the hybrid $\mathsf{Until}/\mathsf{Since}$ language. The complexity of this language over transitive frames and transitive trees, respectively, has been open. PSPACE-completeness over linear frames is known from [13]. We want to find out at which exact requirements to the frame classes the decrease from EXP to PSPACE takes place.

Undecidability results for languages containing $\downarrow$ originate from [7, 15]. The strongest such result, namely for a restricted fragment of the $\downarrow$ language, is given in [2]. In recent work [31], it was shown that decidability of the $\downarrow$ language can be regained under certain restrictions on the frame classes. It is possible that transitivity is another property under which the $\downarrow$ language can be "tamed".

Moreover, we have already observed that over transitive trees and linear orders, the $\downarrow$ operator on its own is useless. Hence satisfiability over these frame classes is the same as for modal logic, namely complete for PSPACE and NP, respectively. This makes it more likely that we can "tame" $\downarrow$ over transitive frames. But if so, to what extent? What happens if we allow for interactions of $\downarrow$ with @ or the backward modality?

New Road-Map Pages. This paper establishes two groups of complexity results for hybrid languages over transitive frames and transitive trees.

First we examine satisfiability of the hybrid $\downarrow$ language. Our most surprising result is the "taming" of this language over transitive frames: the satisfiability problem is NEXP-complete. This high level of complexity is retained even over complete frames (clusters). We also show that enriching the language by the backward-looking modality P or the @ operator leads to undecidability in the case of transitive frames. Over transitive trees, the situation is different. Decidability for even the richest $\downarrow$ language is easy to see, but it is nonelementary, as we will show.

As a second step, we consider satisfiability over transitive frames and transitive trees for the hybrid Until/Since-E language. We establish EXP-hardness for not more than the modal language with Until only. This is matched by an EXP upper bound for the full language in the case of transitive trees. As for transitive frames, we give a 2EXP upper bound.

Table 1 gives an overview of the satisfiability problems considered in this paper (marked bold) and visualizes how our complexity results arrange into a collection of previously known results. It makes use of the denotation of hybrid languages introduced in Section 2. Complexity classes without addition stand for completeness results; "nonel." stands for "nonelementarily decidable". The work from which the results originate, is cited. Conclusions from surrounding results are abbreviated by "c.". The question mark stands for an open question, but decidability follows from the last result in that column. As for $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U,S}}$ over transitive frames and transitive trees, EXP-hardness even holds for $\mathcal{ML}_{\mathsf{U}}$.

hybrid lang.	complexity over arbitrary frames	complexity over transitive frames	complexity over transitive trees	complexity over linear orders
$\mathcal{HL}^{@}$	PSPACE [2]	PSPACE [3]	PSPACE [3, c.]	NP [3, c.]
$\mathcal{HL}_{\mathsf{F,P}}$	EXP [2]	EXP [3]	PSPACE [3]	NP [3]
$\mathcal{HL}^{\mathsf{E}}_{\mathsf{F,P}}$	EXP [3]	EXP [3]	PSPACE [3]	NP [3]
$\mathcal{HL}^{\mathsf{E}}_{\mathsf{U,S}}$	EXP [3]	**in 2EXP** (Th. 14), **EXP-hard** (Th. 12)	**EXP** (Th. 12,15)	PSPACE-hard [26]
$\mathcal{HL}^{\downarrow}$	coRE [2]	**NEXP** (Th. 1)	PSPACE [3]	NP [13]
$\mathcal{HL}^{\downarrow,@}$	coRE [2]	**coRE** (Th. 10)	**nonel.** (Th. 11)	?
$\mathcal{HL}^{\downarrow}_{\mathsf{F,P}}$	coRE [2, c.]	**coRE** (Th. 10)	**nonel.** (Th. 11)	nonel. [13, c.]
$\mathcal{HL}^{\downarrow,@}_{\mathsf{F,P}}$	coRE [2, c.]	**coRE** (c.)	**nonel.** (Th. 11)	nonel. [13]

Table 1. An overview of complexity results for hybrid logics.

Legend. This paper is organized as follows. In Section 2, we give all necessary definitions and notations of modal and hybrid logic as well as an overview of our results in the context of previous work. We present the decidability and undecidability results for the hybrid $\downarrow$ languages in Sections 3 and 4. The hybrid Until/Since language is examined in section 5, and Section 6 ends the paper with some concluding remarks.

2 Modal and Hybrid Logic

We define the basic concepts and notations of modal and hybrid logic that are relevant for this paper. The fundamentals of modal logic can be found in [6]; those of hybrid logic in [2, 5].

Modal Logic. Let PROP be a countable set of *propositional atoms*. The language $\mathcal{ML}$ of modal logic is the set of all formulae of the form $\varphi ::= p \mid \neg\varphi \mid$

$\varphi \wedge \varphi' \mid \Diamond\varphi$, where $p \in$ PROP. We use the well-known abbreviations $\vee$, $\rightarrow$, $\leftrightarrow$, $\top$ ("true"), and $\bot$ ("false"), as well as $\Box\varphi := \neg\Diamond\neg\varphi$.

A *(Kripke) model* is a triple $\mathcal{M} = (M, R, V)$, where M is a nonempty set of *states*, $R \subseteq M \times M$ is a binary relation — the *accessibility relation* —, and $V : \text{PROP} \rightarrow \mathfrak{P}(M)$ is a function — the *valuation function*. The structure $\mathcal{F} = (M, R)$ is called a *frame*. Given a model $\mathcal{M} = (M, R, V)$ and a state $m \in M$, the *satisfaction relation* is defined by

$$\begin{array}{lll} \mathcal{M}, m \models p & \text{iff} & m \in V(p),\ p \in \text{PROP}, \\ \mathcal{M}, m \models \neg\varphi & \text{iff} & \mathcal{M}, m \not\models \varphi, \\ \mathcal{M}, m \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, m \models \varphi \ \&\ \mathcal{M}, m \models \psi, \\ \mathcal{M}, m \models \Diamond\psi & \text{iff} & \exists n \in M(mRn \ \&\ \mathcal{M}, n \models \psi). \end{array} \tag{2.1}$$

A formula φ is *satisfiable* if there exist a model $\mathcal{M} = (M, R, V)$ and a state $m \in M$, such that $\mathcal{M}, m \models \varphi$. If all states from $\mathcal{M}$ satisfy φ, we write $\mathcal{M} \models \varphi$ and say that φ is *globally satisfied* by $\mathcal{M}$.

Temporal Logic. The language of temporal logic (tense logic) is the set of all formulae of the form $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \mathsf{F}\varphi \mid \mathsf{P}\varphi$, where $p \in$ PROP. It is common practice to use the abbreviations $\mathsf{G}\varphi := \neg\mathsf{F}\neg\varphi$ and $\mathsf{H}\varphi := \neg\mathsf{P}\neg\varphi$. The operator F replaces $\Diamond$, hence satisfaction for F-formulae is defined as in (2.1). In the case of P-formulae, the term mRn must be replaced by nRm.

Whenever one wants to speak not only of states accessible from the actual state, but also of states "between" the actual and some accessible state, one can make use of the binary operators U ("until") and S ("since"), for which satisfaction is defined by

$$\begin{array}{l} \mathcal{M}, m \models \mathsf{U}(\varphi, \psi) \text{ iff } \exists n \in M\big(mRn \ \&\ \mathcal{M}, n \models \varphi \ \&\ \forall s \in M(mRsRn \Rightarrow \mathcal{M}, s \models \psi)\big), \\ \mathcal{M}, m \models \mathsf{S}(\varphi, \psi) \text{ iff } \exists n \in M\big(nRm \ \&\ \mathcal{M}, n \models \varphi \ \&\ \forall s \in M(nRsRm \Rightarrow \mathcal{M}, s \models \psi)\big). \end{array}$$

The U/S language is strictly stronger than the basic temporal language in the sense that F and P can be expressed by U and S $\big($e.g. $\mathsf{F}\varphi = \mathsf{U}(\varphi, \top)\big)$, but not vice versa.

In [3], a variant of the U/S operators, U^+ and S^+, is introduced. Satisfaction for U^+ (analogously for S^+) is defined by

$$\begin{array}{l} \mathcal{M}, m \models \mathsf{U}^+(\varphi, \psi) \\ \quad \text{iff } \exists n \in M\big(mRn \ \&\ \mathcal{M}, n \models \varphi \ \&\ \forall s \in M(mR^+sR^+n \Rightarrow \mathcal{M}, s \models \psi)\big), \end{array} \tag{2.2}$$

where R^+ is the transitive closure of R. With the help of U^+ and S^+, the authors of [3] "simulated" transitive frames syntactically. We go a step further and define another modification, U^{++} and S^{++}, with the satisfaction relation from (2.2), where the last remaining term mRn is replaced by mR^+n. This temporal language is an even closer simulation of transitivity, as we will see in Section 5.

Hybrid Logic. As indicated in the previous section, *the* hybrid language does not exist. Rather there are several extensions of the modal language allowing for explicit references to states and therefore being called hybrid. We introduce the richest of those hybrid languages that will interest us in this paper. The definitions and notations are taken from [2, 3].

Let NOM be a countable set of *nominals*, SVAR be a countable set of *state variables*, and ATOM = PROP $\cup$ NOM $\cup$ SVAR. It is common practice to write propositional atoms as $p, q, \ldots$, nominals as $i, j, \ldots$, and state variables as $x, y, \ldots$ The *full hybrid language* $\mathcal{HL}^{\downarrow,@}$ is the set of all formulae of the form $\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \Diamond\varphi \mid @_t\varphi \mid {\downarrow}x.\varphi$, where $a \in$ ATOM, $t \in$ NOM $\cup$ SVAR, and $x \in$ SVAR.

A formula is called *pure* iff it contains no propositional atoms; *nominal-free* iff it contains no nominals; and a *sentence* iff it contains no free state variables. (*Free* and *bound* are defined as usual; the only binding operator here is $\downarrow$.)

A *hybrid model* is a Kripke model with the valuation function V extended to PROP $\cup$ NOM, where for all $i \in$ NOM, $|V(i)| = 1$. Whenever it is clear from the context, we will omit the word "hybrid" when referring to models. In order to evaluate $\downarrow$-formulae, an *assignment* g : SVAR $\to M$ *for* $\mathcal{M}$ is necessary. Given an assignment g, a state variable x and a state m, *an x-variant g^x_m of g* is defined by $g^x_m(x) = m$ and $g^x_m(x') = g(x')$ for all $x' \neq x$. For any atom a, let $[V, g](a) = \{g(a)\}$ if $a \in$ SVAR, and $V(a)$, otherwise. The satisfaction relation for hybrid formulae is defined by

$$\begin{array}{lll}
\mathcal{M}, g, m \models a & \text{iff} & m \in [V, g](a),\ a \in \text{ATOM}, \\
\mathcal{M}, g, m \models \neg\varphi & \text{iff} & \mathcal{M}, g, m \not\models \varphi, \\
\mathcal{M}, g, m \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, g, m \models \varphi \;\&\; \mathcal{M}, g, m \models \psi, \\
\mathcal{M}, g, m \models \Diamond\varphi & \text{iff} & \exists n \in M(mRn \;\&\; \mathcal{M}, g, n \models \varphi), \\
\mathcal{M}, g, m \models @_t\varphi & \text{iff} & \mathcal{M}, g, n \models \varphi \;\&\; [V, g](t) = \{n\}, \\
\mathcal{M}, g, m \models {\downarrow}x.\varphi & \text{iff} & \mathcal{M}, g^x_m, m \models \varphi.
\end{array}$$

A formula is *satisfiable* if there exist a model $\mathcal{M} = (M, R, V)$, an assignment g for $\mathcal{M}$, and a state $m \in M$, such that $\mathcal{M}, g, m \models \varphi$.

We sometimes use the "somewhere" modality E having the interpretation $\mathcal{M}, g, m \models \mathsf{E}\varphi$ iff $\exists n \in M(\mathcal{M}, g, n \models \varphi)$. In this case, @ is needless, because $@_t\,\varphi$ can be expressed by $\mathsf{E}(t \wedge \varphi)$.

First-order Logic. Modal and hybrid logic can be embedded into fragments of first-order logic (FOL). We will always use the standard notation of FOL.

We will make use of certain fragments of FOL and denote them in the style of [10]: $[\text{all}, (u, 1)]$, where $u \in \omega$. This notation stands for the fragment without equality, without function symbols, and with no other relation symbols than one binary and u unary ones. We denote the satisfiability problem for such a fragment by $[\text{all}, (u, 1)]$-SAT and $[\text{all}, (u, 1)]$-trans-SAT, where the latter requires that the binary relation symbol is interpreted by a transitive relation.

The *Standard Translation* ST [31] embeds hybrid logic into FOL and consists of two functions ST_x and ST_y defined recursively. Since ST_y is obtained from ST_x by exchanging x and y, we only give ST_x here.

$$
\begin{array}{rclrcl}
\mathrm{ST}_x(p) & = & P(x), & \mathrm{ST}_x(\Diamond\varphi) & = & \exists y\big(xRy \wedge \mathrm{ST}_y(\varphi)\big), \\
\mathrm{ST}_x(t) & = & t{=}x, & \mathrm{ST}_x(@_t\varphi) & = & \exists y\big(y{=}t \wedge \mathrm{ST}_y(\varphi)\big), \\
\mathrm{ST}_x(\neg\varphi) & = & \neg\,\mathrm{ST}_x(\varphi), & \mathrm{ST}_x(\downarrow v.\varphi) & = & \exists v\big(x{=}v \wedge \mathrm{ST}_x(\varphi)\big), \\
\mathrm{ST}_x(\varphi\wedge\psi) & = & \mathrm{ST}_x(\varphi)\wedge\mathrm{ST}_x(\psi), & \mathrm{ST}_x(\mathsf{E}\varphi) & = & \exists y\big(\,\mathrm{ST}_y(\varphi)\big),
\end{array}
$$

where $p \in \mathrm{PROP}$, $t \in \mathrm{NOM} \cup \mathrm{SVAR}$, and $v \in \mathrm{SVAR}$.

Properties of Models and Frames. Let $\mathcal{M} = (M, R, V)$ be a (Kripke or hybrid) model with the underlying frame $\mathcal{F} = (M, R)$. By R^+ we denote the transitive closure of R. For any subset $M' \subseteq M$, we write $R\restriction_{M'}$ and $V\restriction_{M'}$ for the restrictions of R and V to M'. We will refer to *transitive frames* or *linear frames* whenever we mean frames whose accessibility relation is transitive or a linear order, respectively. A *linear order* is an irreflexive, transitive, and trichotomous $\big(\forall xy(xRy \text{ or } x{=}y \text{ or } yRx)\big)$ relation. The frame $\mathcal{F}$ is a *tree* iff it is acyclic and connected, and every point has at most one R-predecessor. A *transitive tree* is any (M, R^+), where (M, R) is a tree.

Satisfiability Problems. Whenever we leave one or more operators out of the hybrid language, we omit the according superscript of $\mathcal{HL}$. If we proceed to a hybrid tense language, we add the suitable temporal operator(s) as subscript(s) to $\mathcal{HL}$. Analogously, when equipping the modal language with additional operators, we add them as sub- or superscripts to $\mathcal{ML}$.

For any hybrid language $\mathcal{HL}_y^x$, the *satisfiability problem* $\mathcal{HL}_y^x$-SAT is defined as follows: Given a formula $\varphi \in \mathcal{HL}_y^x$, do there exist a hybrid model $\mathcal{M}$, an assignment g for $\mathcal{M}$, and a state $m \in M$ such that $\mathcal{M}, g, m \models \varphi$? If $\downarrow$ is not in the considered language, then the assignment g may be left out of this formulation. If we only ask for *transitive models* (or *transitive trees* or *linear models*, respectively) satisfying φ, then we speak of $\mathcal{HL}_y^x$-trans-SAT (or $\mathcal{HL}_y^x$-tt-SAT, or $\mathcal{HL}_y^x$-lin-SAT, respectively). As an example for our notation, the satisfiability problem over transitive frames for the hybrid temporal $\downarrow$ language is denoted by $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-trans-SAT.

3 Deciding $\mathcal{HL}^{\downarrow}$ over Transitive Frames

In [2] Areces, Blackburn, and Marx proved that the downarrow operator $\downarrow$ turns the satisfiability problem for hybrid logics undecidable in general, even if no interaction with @ or P is allowed. We prove that undecidability vanishes if frames are required to be transitive.

Theorem 1 *The satisfiability problem for $\mathcal{HL}^{\downarrow}$ over transitive frames is complete for* NEXP.

Before we start with the proof we have a first look at $\mathcal{HL}^{\downarrow}$ over transitive frames. Obviously, it has no finite model property. E.g., the following sentence requires a model containing an infinite chain of states labeled p.

$$p \wedge \Diamond p \wedge \Box\Diamond p \wedge \Box \downarrow x.\neg\Diamond x$$

Neither is it always possible to find a model that is a transitive tree. But, in some way, we can get close to this. Although our models may contain cycles, transitivity ensures that all states in a cycle are pairwise connected. I.e., the subframe consisting of these states is complete. Therefore, we can view a model as consisting of maximal complete subframes and single states, that are connected in a transitive but acyclic fashion.

For every transitive model $\mathcal{M} = (M, R, V)$ we define $B(\mathcal{M}) = (M', R', V')$ to be the model defined as follows. First, we replace each maximal complete subframe with a single vertex. Second, we unravel the resulting structure into a (potentially infinite) transitive tree T. Then we replace each vertex of T by (a copy of) the clique of $\mathcal{M}$ from which it is derived. We will refer to the resulting model as the *block tree* of $\mathcal{M}$. Note that a block tree is not a tree but its clique frame is a transitive tree.

In the following we are often interested in the underlying tree structure of a block tree and refer to the cliques of a block tree as *nodes*. For a state s, we denote its node by u_s. We say that a node v *is below* a node u if the states of v are reachable from the states in u (but not vice versa) and a node v is a *child* of a node u, if v is below u but there is no node w below u and above v.

Likewise we use the terms tree, subtree, and leaf for block tree, sub block tree and leaf clique, respectively.

We have to be careful about how to treat nominals when unraveling a model. If $V(i) = \{s\}$ for a nominal i and a state $s \in M$, we define $V'(i)$ to be the set of states from M', that are copies via the unraveling of s. Therefore, $B(\mathcal{M})$ is not a model as defined in Section 2 because nominals may hold at more than one state, but by viewing i as a propositional atom it can be treated as a model. The satisfaction relation is not affected and it is therefore easy to see that this transformation preserves satisfaction of $\mathcal{HL}^{\downarrow}$-sentences: The relation associating each state of $\mathcal{M}$ with every copy in $B(\mathcal{M})$ is a quasi-injective bisimulation[3] [8].

Lemma 2 *For every transitive model $\mathcal{M}$ and every $\mathcal{HL}^{\downarrow}$-sentence φ:*

$$\mathcal{M} \models \varphi \iff B(\mathcal{M}) \models \varphi.$$

Note that we can always get a model for φ from $B(\mathcal{M})$ by joining the states labeled with the same nominals, but this model might be different from $\mathcal{M}$.

Before we show how to use this tree-like structure to decide $\mathcal{HL}^{\downarrow}$ over transitive frames, we focus on complete subframes and show that their size can be bounded.

[3] The notion of bisimulation has to be extended by requiring states carrying the same nominal to be related.

3.1 $\mathcal{HL}^{\downarrow}$ over Complete Frames

As complete subframes are a significant part of transitive models for $\mathcal{HL}^{\downarrow}$-sentences, we are now going to study the satisfiability problem of $\mathcal{HL}^{\downarrow}$ over complete frames. The most important result for our purpose is an exponential-size model property of $\mathcal{HL}^{\downarrow}$ over complete frames.

We want to start by giving some insight why this property holds. In complete frames, the accessibility relation does not distinguish different states. Of course, states can be told apart if they are labeled differently by propositions. But the number of different labelings is exponentially bounded in the size of the formula. To use more states, we have to distinguish states labeled equally. This can only be done by assigning names to these states. But the number of states we can distinguish in this way is bounded by the number of different state variables.

While intuition is clear, we can prove this bound by observing that $\mathcal{HL}^{\downarrow}$ over complete frames is equivalent to the *Monadic Class with equality* ($\mathsf{MC}_=$), the fragment of first-order logic with only unary predicates, equality, and no function symbols [10].

This result allows us to transfer complexity results and model properties for $\mathsf{MC}_=$ [10] to $\mathcal{HL}^{\downarrow}$ over complete frames.

Theorem 3 *$\mathcal{HL}^{\downarrow}$ over complete frames has the exponential-size model property and its satisfiability problem is complete for* NEXP.

The lower bound can be transfered directly to the case of transitive frames.

Corollary 4 *The satisfiability problem for $\mathcal{HL}^{\downarrow}$ over transitive frames is hard for* NEXP.

3.2 On Transitive Frames for $\mathcal{HL}^{\downarrow}$

Let us summarize what we have seen so far. For every $\mathcal{HL}^{\downarrow}$-formula satisfiable over transitive frames, instead of a transitive model we can consider its block tree. The size of the cliques in the block tree can be exponentially bounded in the size of the formula by Corollary 3.

The algorithm for testing $\mathcal{HL}^{\downarrow}$-satisfiability will essentially guess a model and verify that it is correct. As there is no finite model property, all models might be infinite. Nevertheless, we will show that, if the formula is satisfiable, there is always a model with a regular structure in which certain finite patterns are repeated infinitely often. This will allow us to find a finite representation of such a model.

To this end, Definition 5 captures the information about a state of a block tree that will be needed for the following. Intuitively, the φ-type of a state captures the information needed about its subtree in order to evaluate a subformula of a given formula φ at this state. Here, $\psi[free/\bot]$ is the sentence obtained from ψ by replacing every free variable by $\bot$ and $sub(\varphi)$ is the set of all subformulae of φ.

Definition 5 *Let φ be a $\mathcal{HL}^{\downarrow}$-sentence and $\mathcal{B} = (M, R, V)$ a block tree model which is a model of φ. The* φ-type *of a state $s \in M$ is the set of all sentences from $\{\psi[free/\bot] \mid \Diamond\psi \in sub(\varphi)\}$ that hold at some state in the subtree rooted at s.*

Note that states in the same clique have the same φ-type. Therefore, we can speak of the φ-type of a node. The type of a node is always a superset of the types of its children. More precisely, it is always the union of the types of the children together with the set of relevant formulae which hold in the node itself.

When evaluating a subformula of a $\mathcal{HL}^{\downarrow}$-sentence φ at some state s of a block tree, all we need to know about states strictly below u_s are the φ-types of the children of u_s. I.e., we can replace subtrees below u_s by subtrees of the same φ-type. In the following lemma, for a block tree $\mathcal{B}$ and two states s_1, s_2, $\mathcal{B}[s_1/s_2]$ denotes the block tree resulting from $\mathcal{B}$ by replacing the subtree rooted at s_1 by the subtree rooted at s_2. The result of this substitution is again a block tree.

Lemma 6 *Let φ be a $\mathcal{HL}^{\downarrow}$-sentence, $\mathcal{B} = (M, R, V)$ a block tree model of φ and s_1 and s_2 states of $\mathcal{M}$ such that there is a path from s_1 to s_2 but not vice versa. For every formula $\psi \in sub(\varphi)$, every state s_3 of $\mathcal{M}$ of the same φ-type as s_2, and every assignment g that maps all free variables in ψ to states in M preceding s_1:*

$$\mathcal{B}, g, s_1 \models \psi \iff \mathcal{B}[u_{s_2}/u_{s_3}], g, s_1 \models \psi.$$

Note that we restricted the choice of g only to those assignments that are really relevant when evaluating the sentence φ.

We can use the previous lemma to get some nice restrictions on the block trees under consideration. E.g., we can assume that for every sentence in the type of a node, there is a witness in the node itself or in one of its children.

Lemma 7 *Let φ be a $\mathcal{HL}^{\downarrow}$-sentence satisfiable over transitive frames. Then there is a block tree model $\mathcal{B}$ for φ, in which*

- *every node has at most $|\varphi|$ children,*
- *for every node u with φ-type t and every $\mathcal{HL}^{\downarrow}$-sentence $\psi \in t$, ψ holds at a state in u or at a state in a child of u, and*
- *on every path from the root, infinite or ending at a leaf, every φ-type occurs only once or infinitely often.*

3.3 Deciding $\mathcal{HL}^{\downarrow}$-SAT over Transitive Frames

We will now finish the proof of Theorem 1 by presenting a nondeterministic algorithm that decides $\mathcal{HL}^{\downarrow}$-SAT over transitive frames in exponential time, basically by guessing and verifying the finite representation of a block tree model for a given $\mathcal{HL}^{\downarrow}$-sentence φ.

Given a block tree $\mathcal{B}$ with the properties of Lemma 7, we get a finite representation as follows. For each path of $\mathcal{B}$ we consider the first node v that has the same type as its parent node u. We replace the subtree below v by a single state labeled with a reference to u. We need to keep v because it might be the only witness for a formula in the φ-type of u (cf. Lemma 7). Clearly, the resulting structure is finite.

By Lemma 6 and Lemma 7, we can get a block tree model from this representation by replacing each reference with the subtree rooted at the referenced node, i.e., essentially by an unraveling.

Due to Lemma 6, the size of the representation can be reduced even further. If there are two nodes u and v of the same φ-type which are not on the same path and both are the first node of their type on their path from the root, we can replace the subtree rooted at v with the subtree of u. I.e., whenever two nodes have the same φ-type, we can assume that their generated subtrees are equal. We have to check them only once.

Such a representation can be described by a structure $(M \dot{\cup} C, R, V)$ such that the states in C have no outgoing edges, and a function f from C to M. Note that a state in C is a node of its own, in fact a leaf, and cannot be in the same complete subframe as a state in M. A state $s \in C$ stands for a repetition respectively duplication of the subtree rooted at $f(s)$, including states from C. This causes infinite repetition if s is below $f(s)$.

This observation can be reflected in our representation by replacing every duplicate with a reference. This causes every type to appear at most twice in the representation, thus the number of nodes is at most exponential.

Summing up, if φ is satisfiable, there is a representation for of a block tree model for φ of size at most exponential in the length of φ. The first step of the algorithm is to guess such a representation (step 1).

In order to obtain an algorithm which tests whether the representations indeed represents a model of φ, we describe how to modify the model checking algorithm MCFULL by Franceschet and de Rijke presented in [12] to do so. First, we deal with the states in C (step 2). Next, the model checking algorithm MCFULL is used on the states in M (step 3). We have to modify this algorithm in two respects. First, it has to use the information guessed for the states in C. Second, it should compute the φ-types of the states in M. To this end, it first evaluates the sentences resulting from subformulae of φ by replacing free variables with $\perp$. We call this modified algorithm MCFULL'. The changes are straightforward.

After running MCFULL' the algorithm has computed for each state the set of formulae that hold at this state. These sets depend on the guesses in Step 2. Therefore, the algorithm has to verify the consistency of these guesses. This can be done by comparing the φ-types of the states in C with the types of the referenced states (step 4).

The last two steps are easy. The algorithm checks that the φ-types are consistent and reject if this is not the case (step 5). Finally, it checks if φ holds at some state in M (step 6).

Our algorithm for $\mathcal{HL}^{\downarrow}$-satisfiability over transitive frames

1: Guess the representation of a block tree.
2: Guess a φ-type for every state in C.
3: Run MCFULL' for the states in M.
4: Compute the φ-type for every state in M referenced by a state in C.
5: Check for every state $s \in C$: $f(s)$ has the same φ-type as s. If not, reject.
6: Accept iff φ holds at some state in M.

Theorem 8 *The algorithm presented above decides $\mathcal{HL}^{\downarrow}$-satisfiability over transitive frames nondeterministically in exponential time.*

Proof. In Section 3.2, we have seen that for every satisfiable sentence φ there is a block tree model as described in Lemma 7. We have also seen how to represent this block tree in a finite manner. The algorithm can guess this representation and the φ-types of the states in C. The computation of the φ-types of the states in M works correctly, because we have a witness for every sentence in the type in our representation. This is by Lemma 7, which ensures that witnesses are in the node of the state or in one of its children. Therefore, we cut below these witnesses when building the finite representation. Consequently, the algorithm will accept.

On the other hand, if the algorithm accepts, it is straightforward to construct a block tree model from the guessed representation. The only critical point for soundness is the verification of the φ-types guessed in Step 2, more precisely, the computation of the φ-types of the states in M. First, the φ-type of some state $s \in M$ contains only sentences that hold at some state of the represented model below s. This can be assured by looking only at states in M and not at states in C. That the φ-type of s contains all sentences that hold below s can be assured by following the links represented by states in C.

The first two steps of the algorithm can be performed in exponential time because the representation is of at most exponential size. That Step 3 runs in exponential time follows from Theorem 4.5 of [12], the truth of which is not affected by our modifications. The time bounds for the other steps follow again from the exponential size bound of the representation. ❑

From Theorem 8 and Corollary 4 we can directly conclude Theorem 1.

4 Richer hybrid $\downarrow$ Logics over Transitive Frames and Transitive Trees

This section is concerned with satisfiability over transitive frames and transitive trees for extensions of $\mathcal{HL}^{\downarrow}$.

Our first point is that we cannot sustain decidability over transitive frames if we enrich $\mathcal{HL}^{\downarrow}$ with @ or the backward looking modality P. We prove undecidability in both cases, proceeding in two steps. First, we show that $[\text{all}, (4,1)]$-trans-SAT is undecidable. This is done by a reduction from $[\text{all}, (0,1)]$-SAT. The undecidability of the latter is a consequence of the undecidability of contained traditional standard classes [10]. The second step consists of reduction from $[\text{all}, (4,1)]$-trans-SAT to $\mathcal{HL}^{\downarrow,@}$-trans-SAT and $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-trans-SAT, respectively. To be more precise, the ranges of these reductions will be the fragments of the respective hybrid languages consisting of all nominal-free sentences.

Lemma 9 $[\text{all}, (4,1)]$-trans-SAT *is undecidable.*

Proof. In order to obtain the required reduction from $[\text{all}, (0,1)]$-SAT, we will transform a (not necessarily transitive) model satisfying α into a transitive one. Simply taking the transitive closure in most cases adds new pairs to the interpretation of the relation and is not sufficient for keeping the information which pairs were in the "old" relation and which pairs were not. This problem does not arise if we instead use a variation of the *zig-zag technique* successfully applied in [3] for a reduction between a modal and a hybrid language. The core idea of this technique is to simulate an R-step $t_1 R t_2$ in the original model $\mathcal{M} = (D, I)$ by a zig-zag transition in a model $\mathcal{M}' = (D', I')$, where $I'(R)$ is transitive, as shown in Figure 1.

We define a translation function $(\cdot)^t$ using four extra predicate symbols $0, 1, 2, 3$ as follows.

$$\begin{aligned}
(xRy)^t &= \exists abc\, \big(xRa \wedge bRa \wedge bRc \wedge yRc \\
&\qquad \wedge 0(x) \wedge 1(a) \wedge 2(b) \wedge 3(c) \wedge 0(y)\big), \\
(\neg\alpha)^t &= \neg(\alpha^t), \\
(\alpha \wedge \beta)^t &= \alpha^t \wedge \beta^t, \\
(\exists x\, \alpha)^t &= \exists x\, \big(0(x) \wedge \alpha^t\big).
\end{aligned}$$

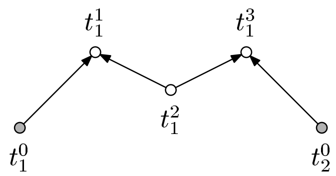

Fig. 1. A zig-zag transition.

The translation of the xRy-atoms exactly reflects the shown zig-zag transition. It is straightforward to prove the following claim: *For each formula α, α is satisfiable iff $f(\alpha)$ is satisfiable in some model that interprets R by a transitive relation.* Since $(\cdot)^t$ is an appropriate (even polynomial-time) reduction function, we have established undecidability for $[\text{all}, (4,1)]$-trans-SAT. ❑

Theorem 10 $\mathcal{HL}^{\downarrow,@}$-trans-SAT *and* $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-trans-SAT *are undecidable.*

Proof. We reduce $[\text{all}, (4,1)]$-trans-SAT to $\mathcal{HL}^{\downarrow,@}$-trans-SAT and $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-trans-SAT, respectively, invoking a spypoint argument (see [7, 2]). A spypoint is a state s of a hybrid model that sees all other states and is named by a fresh nominal i. Since our reduction will not make use of any nominals, we can establish this undecidability result for the nominal-free fragments of the hybrid languages in question. We simply treat i as a state variable and bind it to s.

We first treat the case of $\mathcal{HL}^{\downarrow,@}$ and define a translation function $(\cdot)^t$ from the first-order fragment to $\mathcal{HL}^{\downarrow,@}$ by $(xRy)^t = @_x \Diamond y$, $\big(P(x)\big)^t = @_x\, p$, $(\neg\alpha)^t = \neg(\alpha^t)$, $(\alpha \wedge \beta)^t = \alpha^t \wedge \beta^t$, and $(\exists x\, \alpha)^t = @_i \Diamond {\downarrow} x. \alpha^t$. The (polynomial) reduction function f is defined by $f(\alpha) = {\downarrow} i. (\neg\Diamond i \wedge \Diamond \alpha^t)$.

It is straightforward to show that each formula α is satisfiable iff $f(\alpha)$ is satisfiable, and to adopt the translation function $(\cdot)^t$ to the case of $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$. ❑

The second point of this section is that over transitive trees, where decidability of $\mathcal{HL}^{\downarrow}$ is trivial, even the extension $\mathcal{HL}^{\downarrow,@}_{\mathsf{F},\mathsf{P}}$ is decidable. This is an immediate consequence of the decidability of the monadic second-order theory of the countably branching tree, SωS, [10]. However, we have to face a nonelementary lower bound in both cases $\mathcal{HL}^{\downarrow,@}$ and $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$, which is obtained by a reduction from the nonelementarily decidable $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-lin-SAT [13].

Theorem 11 $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-tt-SAT, $\mathcal{HL}^{\downarrow,@}$-tt-SAT, *and* $\mathcal{HL}^{\downarrow,@}_{\mathsf{F},\mathsf{P}}$-tt-SAT *are nonelementary decidable.*

5 Hybrid Until/Since Logic over Transitive Frames and Transitive Trees

In this section, we will consider $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-trans-SAT and $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-tt-SAT. In [3] it was shown that $\mathcal{HL}^{@}_{\mathsf{U},\mathsf{S}}$-SAT is EXP-complete.

As for the lower bound, we establish a result as general as possible, namely EXP-hardness of $\mathcal{ML}_{\mathsf{U}}$-trans-SAT and $\mathcal{ML}_{\mathsf{U}}$-tt-SAT.

Theorem 12 $\mathcal{ML}_{\mathsf{U}}$-trans-SAT *and* $\mathcal{ML}_{\mathsf{U}}$-tt-SAT *are* EXP-*hard.*

Proof. We will reduce the *global* satisfiability problem for $\mathcal{ML}$ to both our (local) problems $\mathcal{ML}_{\mathsf{U}}$-trans-SAT and $\mathcal{ML}_{\mathsf{U}}$-tt-SAT using the same reduction function. The global satisfiability problem is defined by $\mathcal{ML}$-GLOBSAT $= \{\varphi \in \mathcal{ML} \mid \varphi$ is true in *all* states of some Kripke model $\mathcal{M}\}$. Its EXP-completeness is a direct consequence of the EXP-completeness of $\mathcal{ML}^{\mathsf{E}}$-SAT [28].

It may seem difficult to try reducing this problem over *arbitrary* frames to our satisfiability problem over *transitive* frames. The critical point lies in making a non-transitive model transitive: taking the transitive closure of its relation forces us to add new accessibilities that would disturb satisfaction of $\neg\Diamond$-formulae. Fortunately though, the U operator can make us distinguish the accessibilities from the original model from those that have been added to make the relation transitive. Hence, a translation of $\Diamond\varphi$ should demand: "Make sure that the actual state sees a state in which the translation of φ holds, and that there is no state in between." This translates as $\mathsf{U}(\varphi^t, \bot)$ into the modal language.

To construct the required reduction, we define a translation function $(\cdot)^t : \mathcal{ML} \to \mathcal{ML}_{\mathsf{U}}$ by $p^t = p$, $p \in$ PROP; $(\varphi \wedge \psi)^t = \varphi^t \wedge \psi^t$; $(\neg\varphi)^t = \neg(\varphi^t)$; and $(\Diamond\varphi)^t = \mathsf{U}(\varphi^t, \bot)$. Using $(\cdot)^t$, we construct a reduction function $f : \mathcal{ML} \to \mathcal{ML}_{\mathsf{U}}$ — that is clearly computable in polynomial time — via $f(\varphi) = \varphi^t \wedge \Box\varphi^t$. It is straightforward to prove the following two claims for each $\varphi \in \mathcal{ML}$.

(1) If $\varphi \in \mathcal{ML}$-GLOBSAT, then $f(\varphi) \in \mathcal{ML}_{\mathsf{U}}$-tt-SAT.

(2) If $f(\varphi) \in \mathcal{ML}_{\mathsf{U}}$-trans-SAT, then $\varphi \in \mathcal{ML}$-GLOBSAT. ❑

The upper bounds for $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-trans-SAT and $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-tt-SAT require separate treatment. As for $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-trans-SAT, we use an embedding into an appropriate fragment of first-order logic. In order to eliminate transitivity, we "simulate" this semantic property by syntactic means, namely using the operators U^{++} and S^{++} defined in Section 2.

Lemma 13 *For any* $X \subseteq \{@, \mathsf{E}\}$, $\mathcal{HL}^{X}_{\mathsf{U},\mathsf{S}}$-trans-SAT *and* $\mathcal{HL}^{X}_{\mathsf{U}^{++},\mathsf{S}^{++}}$-SAT *are polynomially reducible to each other.*

Now it is not difficult anymore to obtain a 2EXP upper bound for $\mathcal{HL}^{@}_{\mathsf{U},\mathsf{S}}$-trans-SAT by an embedding into the loosely μ-guarded fragment μLGF of first-order logic whose satisfiability problem is 2EXP-complete [16]. Only the E operator requires a more careful analysis.

Theorem 14 $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-trans-SAT *is in* 2EXP.

To show that $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-tt-SAT is in EXP, too, we use an embedding into $\mathcal{PDL}_{\mathrm{tree}}$, the propositional dynamic logic for sibling-ordered trees introduced in [20, 21]. Finite, node-labelled, sibling-ordered trees are the logical abstraction of XML (eXtensible Markup Language) documents. In [1], it was shown that satisfiability of $\mathcal{PDL}_{\mathrm{tree}}$ formulae at the root of finite trees ($\mathcal{PDL}_{\mathrm{tree}}$-SAT) is decidable in EXP.

Theorem 15 $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-tt-SAT *is in* EXP.

6 Conclusion

We have established two groups of complexity results for hybrid logics over two temporally relevant frame classes, i. e. that of transitive frames and that of transitive trees.

First we have "tamed" $\mathcal{HL}^{\downarrow}$ over transitive frames showing that $\mathcal{HL}^{\downarrow}$-trans-SAT is NEXP-complete. We have shown that, in contrast, $\mathcal{HL}^{\downarrow,@}$-trans-SAT and $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-trans-SAT are undecidable. Over transitive trees, three enrichments of $\mathcal{HL}^{\downarrow}$ are decidable, albeit nonelementarily, namely $\mathcal{HL}^{\downarrow,@}$-tt-SAT, $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-tt-SAT, and $\mathcal{HL}^{\downarrow,@}_{\mathsf{F},\mathsf{P}}$-tt-SAT.

In the second part of our work, we established an EXP lower bound for $\mathcal{ML}_{\mathsf{U}}$-trans-SAT and $\mathcal{ML}_{\mathsf{U}}$-tt-SAT and matched the latter with an EXP upper bound for $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-tt-SAT. This is the same complexity as for satisfiability over *arbitrary* frames for the same language. As for $\mathcal{HL}^{\mathsf{E}}_{\mathsf{U},\mathsf{S}}$-trans-SAT, we have given a 2EXP upper bound. We conjecture EXP-completeness.

Over linear frames, the complexity of hybrid U/S logic is still open. As a special case, satisfiability of $\mathcal{HL}^{@}_{\mathsf{U},\mathsf{S}}$ over $(\mathbb{N}, >)$ is known to be PSPACE-complete [13]. Furthermore, in [13] it was shown that $\mathcal{ML}_{\mathsf{U}}$ is PSPACE-complete over general linear time.

Another open question is the complexity of $\mathcal{HL}^{\downarrow,@}$-lin-SAT and $\mathcal{HL}^{\downarrow}_{\mathsf{F},\mathsf{P}}$-lin-SAT (see the question marks in Table 1). Merely decidability follows from decidability of $\mathcal{HL}^{\downarrow,@}_{\mathsf{F},\mathsf{P}}$-lin-SAT [13].

Acknowledgments

We thank Balder ten Cate, Massimo Franceschet, Maarten Marx, and the anonymous referees for helpful suggestions, discussions, and comments.

References

1. Afanasiev, L., P. Blackburn, I. Dimitriou, B. Gaiffe, E.Goris, M. Marx and M. de Rijke, *PDL for ordered trees*, Journal of Applied Non-Classical Logics **15** (2005), pp. 115–135.
2. Areces, C., P. Blackburn and M. Marx, *A road-map on complexity for hybrid logics*, in: J. Flum and M. Rodríguez-Artalejo, editors, *CSL*, Lecture Notes in Computer Science **1683** (1999), pp. 307–321.
3. Areces, C., P. Blackburn and M. Marx, *The computational complexity of hybrid temporal logics*, Logic Journal of the IGPL **8** (2000), pp. 653–679.
4. Areces, C., P. Blackburn and M. Marx, *Hybrid logics: Characterization, interpolation and complexity*, J. Symb. Log. **66** (2001), pp. 977–1010.
5. Blackburn, P., *Representation, reasoning, and relational structures: a hybrid logic manifesto*, Logic Journal of the IGPL **8** (2000), pp. 339–365.
6. Blackburn, P., M. de Rijke and Y. Venema, "Modal Logic," Cambridge Tracts in Theoretical Computer Science **53**, Cambridge University Press, 2001.
7. Blackburn, P. and J. Seligman, *Hybrid languages*, Journal of Logic, Language and Information **4** (1995), pp. 251–272.
8. Blackburn, P. and J. Seligman, *What are hybrid languages?*, in: M. Kracht, M. de Rijke, H. Wansing and M. Zakharyaschev, editors, *Advances in Modal Logic*, CSLI Publications **1**, Stanford University, 1998 pp. 41–62.
9. Blackburn, P. and M. Tzakova, *Hybrid languages and temporal logic*, Logic Journal of the IGPL **7** (1999), pp. 27–54.
10. Börger, E., E. Grädel and Y. Gurevich, "The Classical Decision Problem," Perspectives of Mathematical Logic, Springer-Verlag, 1997.
11. Dawar, A. and M. Otto, *Modal characterisation theorems over special classes of frames*, in: *LICS* (2005), pp. 21–30.
12. Franceschet, M. and M. de Rijke, *Model checking hybrid logics (with an application to semistructured data)*, Journal of Applied Logic (2005).
13. Franceschet, M., M. de Rijke and B.-H. Schlingloff, *Hybrid logics on linear structures: Expressivity and complexity*, in: *TIME* (2003), pp. 166–173.
14. Ganzinger, H., C. Meyer and M. Veanes, *The two-variable guarded fragment with transitive relations*, in: *Logic in Computer Science*, 1999, pp. 24–34.
15. Goranko, V., *Hierarchies of modal and temporal logics with reference pointers*, Journal of Logic, Language and Information **5** (1996), pp. 1–24.
16. Grädel, E. and I. Walukiewicz, *Guarded fixed point logic*, in: *LICS*, 1999, pp. 45–54.
17. Immerman, N., A. M. Rabinovich, T. W. Reps, S. Sagiv and G. Yorsh, *The boundary between decidability and undecidability for transitive-closure logics*, in: J. Marcinkowski and A. Tarlecki, editors, *CSL*, Lecture Notes in Computer Science **3210** (2004), pp. 160–174.

18. Kieronski, E., *EXPSPACE-complete variant of guarded fragment with transitivity*, in: H. Alt and A. Ferreira, editors, *STACS*, Lecture Notes in Computer Science **2285** (2002), pp. 608–619.
19. Kieronski, E., *The two-variable guarded fragment with transitive guards is 2EXPTIME-hard*, in: A. D. Gordon, editor, *FoSSaCS*, Lecture Notes in Computer Science **2620** (2003), pp. 299–312.
20. Kracht, M., *Syntactic codes and grammar refinement*, Journal of Logic, Language and Information **4** (1995), pp. 41–60.
21. Kracht, M., *Inessential features*, in: C. Retoré, editor, *LACL*, Lecture Notes in Computer Science **1328** (1996), pp. 43–62.
22. Ladner, R. E., *The computational complexity of provability in systems of modal propositional logic*, SIAM J. Comput. **6** (1977), pp. 467–480.
23. Ono, H. and A. Nakamura, *On the size of refutation Kripke models for some linear modal and tense logics*, Studia Logica **34** (1980), pp. 325–333.
24. Papadimitriou, C. H., "Computational Complexity," Addison-Wesley, 1994.
25. Prior, A., "Past, Present and Future," Oxford University Press, 1967.
26. Reynolds, M., *The complexity of the temporal logic with "until" over general linear time*, J. Comput. Syst. Sci. **66** (2003), pp. 393–426.
27. Sistla, A. P. and E. M. Clarke, *The complexity of propositional linear temporal logics*, J. ACM **32** (1985), pp. 733–749.
28. Spaan, E., "Complexity of modal logics," Ph.D. thesis, ILLC, University of Amsterdam (1993).
29. Spaan, E., *The complexity of propositional tense logics*, in: M. d. Rijke, editor, *Diamonds and Defaults*, Kluwer, 1993 pp. 287–307.
30. Szwast, W. and L. Tendera, *On the decision problem for the guarded fragment with transitivity*, in: *LICS*, 2001, pp. 147–156.
31. ten Cate, B. and M. Franceschet, *On the complexity of hybrid logics with binders*, in: C.-H. L. Ong, editor, *CSL*, Lecture Notes in Computer Science **3634** (2005), pp. 339–354.