J Log Lang Inf (2009) 18:465–491
DOI 10.1007/s10849-009-9088-7

Model Checking for Hybrid Logic

Martin Lange

Published online: 23 April 2009

Abstract We consider the model checking problem for Hybrid Logic. Known algorithms so far are global in the sense that they compute, inductively, in every step the set of all worlds of a Kripke structure that satisfy a subformula of the input. Hence, they always exploit the entire structure. Local model checking tries to avoid this by only traversing necessary parts of the input in order to establish or refute the satisfaction relation between a given world and a formula. We present a framework for local model checking of Hybrid Logic based on games. We show that these games are simple reachability games for ordinary Hybrid Logic and weak Büchi games for Hybrid Logic with operators interpreted over the transitive closure of the accessibility relation of the underlying Kripke frame, and show how to solve these games thus solving the local model checking problem. Since the first-order part of Hybrid Logic is inherently hard to localise in model checking, we give examples, in the end, of how global model checkers can be optimised in certain special cases using well-established techniques like fixpoint approximations and divide-and-conquer algorithms.

Keywords Local model checking · Büchi games

1 Introduction

1.1 Hybrid Logic

There is a well-known opposition between Modal Logic (ML) on one hand and First-Order Logic (FOL) on the other. Modal Logic appeals because of its computational properties. Satisfiability in the modal logic K for example is just PSPACE-complete,

M. Lange (✉)
Department of Computer Science, University of Munich, Oettingenstr. 67, 80538 München, Germany
e-mail: martin.lange@ifi.lmu.de

for some other modal logics it is even NP-complete only, etc. The model checking problem for ML is solvable in polynomial time. All this is connected to the fact that modal logics have a built-in aspect of *locality*. The ML-properties that a world of a Kripke structure has are composed from the properties of its immediate neighbourhood.

FOL is not restricted in this way since, in particular, quantification ranges over all elements of the underlying relational structure. Hence, a world of a Kripke structure can have an FOL-property because some other world that is not in its neighbourhood has another property. Not surprisingly, such operators lead to higher expressive power. For instance, while ML-properties are bisimulation-invariant, i.e. no formula of ML can distinguish bisimilar models, FOL-formulas are well capable of that. However, this increase in expressive power naturally leads to an increase in computational complexity: the model checking problem is PSPACE-complete and the satisfiability problem is non-elementary.

Hence, there is no general preference of one of these logics over the other. It is even easy to imagine that there are cases in which neither ML nor FOL is sufficient because of either lack of expressive power or lack of efficient decision procedures. For such cases one would naturally try to combine ML and FOL in a way that retains the good properties of both. One such approach leads to Hybrid Logic (HL) which incorporates into Modal Logic (or better: Temporal Logic) certain first-order features (Bull 1970; Passy and Tinchev 1991).

This logic has attracted a lot of interest in itself (Goranko 1996; Areces et al. 2000; ten Cate 2005; Areces and ten Cate 2006), studying in particular its proof theory and its model theory. On the other hand, Hybrid Logic also has found applications in many diverse areas, for example in artificial intelligence and knowledge representation because it is closely related to description logics (Areces 2000; Blackburn and Tzakova 1998); in computational linguistics because of its relation to feature logics (Blackburn 1993); as a logic for semi-structured data (Bidoit et al. 2004; Franceschet and de Rijke 2006); etc.

While the main focus in the research on decision procedures traditionally lay on the satisfiability and validity problem, the model checking problem for Hybrid Logic has lately also been found worthy of studying, in particular because of its close relationship to the querying and the constraints evaluation problem for XML data (Franceschet and de Rijke 2006). Franceschet and de Rijke classify the model checking problem for various fragments of Hybrid Logic w.r.t. their computational complexity and achieve various result between polynomial and exponential time. For the upper bounds they present explicit model checking algorithms and analyse their space and time consumption. These algorithms are straight-forward extensions of known algorithms for temporal or propositional dynamic logics. This is certainly sufficient for the complexity classification, but it is fair to ask whether there are "better" algorithms. In program verification for example, where model checking techniques have successfully been used for 25 years by now, these naïve approaches would not be considered state-of-the-art. Franceschet and de Rijke also identify the verification of mobile reactive systems as an application for Hybrid Logic model checking (Franceschet and de Rijke 2003). It is fair to say that such applications require optimised algorithms to stand a chance of being of practical use. The aim of this paper is to provide a framework for such optimised algorithms and to present some ideas how to optimise elsewhere in case where this framework does not perform any better than the naïve algorithms. In order to do so, we first take a step back and discuss some methodologies that have arisen in model checking for program verification.

1.2 Model Checking

Model checking describes the process of determining whether or not a given formula of a given logic is satisfied by a given interpretation of this formula, i.e. whether the latter is a *model* of the formula. It has been observed in the early 80's that this decision problem for certain logics, in particular temporal ones, can be used to decide whether or not a given program satisfies a given specification (Emerson and Clarke 1982; Queille and Sifakis 1982).

The branching time temporal logic CTL was one of the first for which *model checkers*—with the program verification purpose—were designed.[1] Such algorithms usually work in a bottom-up fashion: they inductively compute for every subformula of the input formula the set of all worlds in a Kripke structure which satisfy this subformula. This solves the *global model checking problem*: it eventually computes *all* the worlds in a Kripke structure that satisfy the input formula.

Often, one is content with less information and it would suffice to know for a given world whether or not this one does satisfy the input formula. This is usually called the *local model checking problem*. It should be clear that these two problems are interreducible. The global model checking problem solves the local one by a simple lookup at the end, and local model checking can be carried out for every world of the underlying Kripke structure in order to simulate the global one. It should also be clear that neither of these reductions makes for a very pragmatic approach. Instead, there should be algorithms of both kind if there is a demand for both kinds of model checking problems.

The terms *global* and *local* have also been used in the context of model checking with a slightly different meaning. A model checking *algorithm* is usually called *global* if it works in the way described above, i.e. if it computes in a step the set of all worlds satisfying a subformula. It is *local* if it tries to avoid unnecessary computations of the satisfaction relation between all pairs of worlds and subformulas. A local algorithm tries to find the cause for (un-)satisfaction between a world and a subformula by considering as few other pairs of worlds and subformulas as possible. This is simply a special case of lazy evaluation: for instance, in order to check whether or not a world w satisfies the disjunction $\varphi \vee \psi$, one would first check whether it satisfies φ and consider ψ only if the answer in the first case is negative. A similar approach is of course possible for conjunctions, and maybe also for other constructs in the logic at hand.

[1] Note that CTL is structurally very similar to PDL, and the extension of PDL with Converse (Kozen and Tiuryn 1990) is again very similar to Hybrid Logic as it is presented in Franceschet and de Rijke (2006) and here.

Local algorithms have a distinct advantage over global ones: in most cases there is an unbalance in the sizes of the inputs to a model checking problem. The structure, representing the operational semantics of a concurrent program, a database, etc. is usually large in comparison to the formula representing the correctness specification of a query. It is therefore desirable to have model checking algorithms which operate sparsely on the structure. Local model checking is such an approach which attempts to exploit only a part of the structure necessary to answer the local model checking problem. It was one of the first approaches to tackle the state-space explosion problem in program verification which describes exactly the problem of dealing with large structures (Stirling and Walker 1991; Vergauwen and Lewi 1993; Bhat and Cleaveland 1996).

Locality in a model checking algorithm can be introduced in various ways. One of the most prominent approaches is the framework of games (Stirling 1995). While there are also, e.g., tableau calculi for model checking of various logics, games incorporate the aforementioned idea of lazy evaluation in the presence of duality best. Note that in a disjunction one would try to find a disjunct which can be *proved* while in a conjunction one would try to find a conjunct which can be *refuted*. In a tableau calculus this is often formalised through rules with one premiss for disjunctions and two premisses for conjunctions, and locality is thus only semi-incorporated into the calculus. In order to achieve full locality (here: avoid unnecessary evaluations of conjuncts as well) one would have to additionally localise the proof search procedure. Not that this poses a difficult problem in general, but games may be preferable in this setting because the formalisation of choices between players captures locality at the whole. Moreover, describing a model checking problem abstractly as a game may open up a wide range of game solving algorithms for model checking as well. The most prominent example is model checking for the modal μ-calculus which, for the full logic, is of little practical interest. Nevertheless, since it is equivalent to the problem of solving parity games, the modal μ-calculus is a logic for which several different model checking algorithms exist (Cleaveland 1990; Stevens and Stirling 1998; Vöge and Jurdziński 2000; Jurdziński 2000).

Sometimes, however, structures are so large that local model checking does not help matters much. This is the case when the Kripke structure already requires unfeasibly much space to be stored as a directed labeled graph using adjacency lists or matrices as usual. This has led to the invention of *symbolic model checking* (Burch et al. 1992). There, one uses reduced ordered binary decision diagrams (BDD) to store a relational structure. In some cases such representations can be exponentially smaller. Not surprisingly, one loses access to single worlds in the Kripke structure, and local model checking is simply not possible on such representations. Thus, one has to revert to global algorithms again.

1.3 Outline and Aim of the Paper

In Sect. 2 we introduce Hybrid Logic formally. In Sect. 3 we present a framework for local model checking of Hybrid Logic based on games. We prove correctness of this characterisation and show how such games can be solved in order to do model checking. From a game-theoretic point of view, these games turn out to be very simple. They become slightly less simple when the temporal operators in Hybrid Logic are interpreted not over the accessibility relation of the underlying Kripke structure but over its transitive closure. Section 4 deals with that case by extending the games of Sect. 3, proving them correct and showing how to solve them.

The aim of this paper is to introduce some state-of-the-art techniques into model checking for Hybrid Logic. Locality is dealt with by presenting a game-based framework which enables local model checking for Hybrid Logic. We believe that this may be most useful for the Hybrid Logic applications on semi-structured data, as mentioned above, since XML documents or databases may be assumed to be represented similar to an explicit representation of a Kripke structure. For applications of Hybrid Logic in program verification where models may be represented symbolically, these games are not applicable. One can revert to the naïve and global algorithms presented in Franceschet and de Rijke (2006). It is not difficult to see that the operations they use in order to manipulate sets of worlds can also be carried out on BDD representations of such sets. Nevertheless, there is room for optimisations of these algorithms (in particular on BDDs). In Sect. 5 we turn to the question of improving the naïve global algorithms in special cases in order to be able to speed up these algorithms.

2 Syntax and Semantics of Hybrid Logic

2.1 Hybrid Kripke Structures

Let $\mathsf{Var}_2 = \{X, Y, \ldots\}$ be a countably infinite set of second-order variables (aka propositions), and $\mathsf{Var}_1 = \{x, y, \ldots\}$ a countably infinite set of first-order variables such that $\mathsf{Var}_2 \cap \mathsf{Var}_1 = \emptyset$.

A *Kripke frame* is a pair $\mathcal{F} = (W, R)$ where W is a set of *worlds* and $R \subseteq W \times W$ is a binary relation on it called the *accessibility relation*. We will use infix notation for this relation and write wRv instead of $(w, v) \in R$, as well as $wRuRv$ as an abbreviation for wRu *and* uRv. We will also write wR to denote $\{v \in W \mid wRv\}$ and dually $Rv = \{w \in W \mid wRv\}$. The *size* of a Kripke frame $\mathcal{F} = (W, R)$ is $|\mathcal{F}| := |W| + |R|$.

A *Kripke structure* extends a Kripke frame by an evaluation of the second-order variables. It is a pair $\mathcal{K} = (\mathcal{F}, \rho_2)$ where $\mathcal{F} = (W, R)$ is a Kripke frame and $\rho_2 : \mathsf{Var}_2 \to 2^W$.

A *hybrid Kripke structure* extends a Kripke structure by an evaluation of the first-order variables. It is a triple $\mathcal{K} = (\mathcal{F}, \rho_2, \rho_1)$ where $\mathcal{F} = (W, R)$ is a Kripke frame, ρ_2 as above, and ρ_1 is a function of type $\mathsf{Var}_1 \to W$.

We write Var for $\mathsf{Var}_2 \cup \mathsf{Var}_1$ and—given two interpretations ρ_2 and ρ_1 as above—$\rho := \rho_2 \cup \rho_1$ for the merger of these. Note that it is well-defined because no variable can be both second- and first-order. To be precise: $\rho(X) := \rho_2(X)$ and $\rho(x) := \rho_1(x)$. We will therefore denote a hybrid Kripke structure simply as $(\mathcal{F}, \rho)$ instead of $(\mathcal{F}, \rho_2, \rho_1)$.

A *rooted hybrid Kripke structure* is then a triple $\mathcal{K} = (\mathcal{F}, \rho, w)$ where $\mathcal{F} = (W, R)$ is a Kripke frame, ρ as above, and $w \in W$ is a designated world in it. $\mathcal{K}$ is called *finite* if $|W| < \infty$. Note that, nominally, ρ is still an infinite object since its domain contains infinitely many variables. In the context of model checking, there will always be a formula with finitely many variables only, and it suffices to consider the restriction of ρ to that domain which yields a truely finite object.

Since it is going to be clear from the context whether or not we are considering rooted structures we will simply call them hybrid Kripke structures as well. The motivation for considering these rooted stuctures is the local model checking paradigm.

Given an interpretation ρ of the (first-order) variables, a first-order variable x, and a world w of some Kripke frame $\mathcal{F}$, we write $\rho[x \mapsto w]$ for the interpretation that maps x to w and behaves like ρ on all other arguments: $\rho[x \mapsto w](x) := w$, $\rho[x \mapsto w](y) := \rho(y)$ if $y \neq x$, and $\rho[x \mapsto w](X) := \rho(X)$ in case it is also defined on some second-order variables. The notion $\rho[X \mapsto V]$ for some $X \in \mathsf{Var}_2$ and some $V \subseteq W$ is defined in the same way.

Note that we do not explicitly mention nominals. This is purely for presentational purposes. Nominals are first-order variables with a fixed interpretation in a hybrid Kripke structure. This can easily be modelled by reserving some first-order variable names which are not allowed to be quantified over in a formula.

2.2 Syntax of Hybrid Logic

Let Var_2 and Var_1 be as above. Hybrid Logic extends Temporal Logic by binding and quantification mechanisms. Formulas of Hybrid Logic over $\mathsf{Var} := \mathsf{Var}_2 \cup \mathsf{Var}_1$ are given by the following grammar.

$$\begin{aligned}\varphi ::= {} & \top \mid X \mid \varphi \wedge \varphi \mid \neg\varphi \mid \\ & \mathtt{F}\,\varphi \mid \mathtt{P}\,\varphi \mid \varphi\,\mathtt{U}\,\varphi \mid \varphi\,\mathtt{S}\,\varphi \mid \\ & x \mid @_x\varphi \mid \downarrow x.\varphi \mid \exists x.\varphi\end{aligned}$$

where $X \in \mathsf{Var}_2$ and $x \in \mathsf{Var}_1$. The first line contains the usual definition of Propositional Logic, in the second this is extended to Temporal Logic, and the third completes Hybrid Logic with the aforementioned additional constructs.

Other constructs can be introduced as abbreviations, namely some standard ones from Propositional Logic, $\bot := \neg\top$, $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, $\varphi \to \psi := \neg\varphi \vee \psi$, etc.; from Temporal Logic, $\mathtt{G}\,\varphi := \neg\,\mathtt{F}\,\neg\varphi$, $\mathtt{H}\,\varphi := \neg\,\mathtt{P}\,\neg\varphi$; and from Predicate Logic, $\forall x.\varphi := \neg\exists x.\neg\varphi$, etc.

The *Fischer-Ladner closure* $FL(\varphi)$ is the least set that contains φ and satisfies the following closure properties.

- if $\psi_1 \wedge \psi_2 \in FL(\varphi)$ then $\psi_1, \psi_2 \in FL(\varphi)$,
- if $\neg(\psi_1 \wedge \psi_2) \in FL(\varphi)$ then $\neg\psi_1, \neg\psi_2 \in FL(\varphi)$,
- if $\neg\neg\psi \in FL(\varphi)$ then $\psi \in FL(\varphi)$,
- if $\mathtt{F}\,\psi \in FL(\varphi)$ or $\mathtt{P}\,\psi \in FL(\varphi)$ then $\psi \in FL(\varphi)$,
- if $\neg\,\mathtt{F}\,\psi \in FL(\varphi)$ or $\neg\,\mathtt{P}\,\psi \in FL(\varphi)$ then $\neg\psi \in FL(\varphi)$,
- if $\psi_1\,\mathtt{U}\,\psi_2 \in FL(\varphi)$ or $\psi_1\,\mathtt{S}\,\psi_2 \in FL(\varphi)$ then $\psi_1, \psi_2 \in FL(\varphi)$,
- if $\neg(\psi_1\,\mathtt{U}\,\psi_2) \in FL(\varphi)$ or $\neg(\psi_1\,\mathtt{S}\,\psi_2) \in FL(\varphi)$ then $\neg\psi_1, \neg\psi_2 \in FL(\varphi)$,
- if $@_x\psi \in FL(\varphi)$ or $\downarrow x.\psi \in FL(\varphi)$ or $\exists x.\psi \in FL(\varphi)$ then $\psi \in FL(\varphi)$,
- if $\neg @_x\psi \in FL(\varphi)$ or $\neg \downarrow x.\psi \in FL(\varphi)$ or $\neg\exists x.\psi \in FL(\varphi)$ then $\neg\psi \in FL(\varphi)$.

We write $\mathsf{Var}_1(\varphi)$, resp. $\mathsf{Var}_2(\varphi)$ for the first-, resp. second-order variables occurring in φ, and define $\mathit{Var}(\varphi) := \mathsf{Var}_1(\varphi) \cup \mathsf{Var}_2(\varphi)$. We will also write $\mathit{Var}(\varphi, \psi)$ instead of $\mathit{Var}(\varphi) \cup \mathit{Var}(\psi)$ etc.

An occurrence of a variable x or X in φ is called *free* if it is not under the scope of a quantifier $\exists x$ or $\downarrow x$. It is *bound* if it is not free.

We use the notation $\varphi[\psi/x]$ to denote the simultaneous substitution of the formula ψ for every free occurrence of the variable x in φ.

Finally, we introduce two measures for the structural complexity of a formula φ. The first one is its size, $|\varphi|$ which we define as $|FL(\varphi)|$. Note that the size of the Fischer-Ladner closure is at most twice the syntactical length which may appear to be a better definition of a formula's size. However, the syntactical length can also be exponential in the number of subformulas and, hence, the size of the Fischer-Ladner closure when multiply occurring subformulas are counted separately. It is therefore not such a natural measure indeed.

The second measure in the complexity of model checking Hybrid Logic is the number of distinct bound first-order variables occurring in a formula. Let

$$bv(\varphi) := |\{x \mid x \in \mathsf{Var}_1, x \text{ occurs under the scope of } \exists x \text{ or } \downarrow x \text{ in } \varphi\}|$$

2.3 Semantics of Hybrid Logic

Formulas of Hybrid Logic are interpreted over Kripke frames $\mathcal{F} = (W, R)$ via an interpretation ρ of the second- and first-order variables. The semantics explains under what circumstances a world w of $\mathcal{F}$ satisfies a hybrid formula φ w.r.t. the interpretation ρ, written $\mathcal{F}, w \models_\rho \varphi$ and being defined recursively as follows.

$$\begin{array}{lll}
\mathcal{F}, w \models_\rho \top & & \\
\mathcal{F}, w \models_\rho X & \text{iff} & w \in \rho(X) \\[1ex]
\mathcal{F}, w \models_\rho \varphi \land \psi & \text{iff} & \mathcal{F}, w \models_\rho \varphi \text{ and } \mathcal{F}, w \models_\rho \psi \\[1ex]
\mathcal{F}, w \models_\rho \neg\varphi & \text{iff} & \mathcal{F}, w \not\models_\rho \varphi \\[1ex]
\mathcal{F}, w \models_\rho \mathtt{F}\,\varphi & \text{iff} & \text{there is a } v \in W \text{ with } wRv \text{ and } \mathcal{F}, v \models_\rho \varphi \\[1ex]
\mathcal{F}, w \models_\rho \mathtt{P}\,\varphi & \text{iff} & \text{there is a } v \in W \text{ with } vRw \text{ and } \mathcal{F}, v \models_\rho \varphi \\[1ex]
\mathcal{F}, w \models_\rho \varphi\,\mathtt{U}\,\psi & \text{iff} & \text{there is a } v \in W \text{ with } wRv \text{ and } \mathcal{F}, v \models_\rho \psi \text{ and} \\
 & & \quad \text{for all } u: \text{ if } wRuRv \text{ then } \mathcal{F}, u \models_\rho \varphi \\[1ex]
\mathcal{F}, w \models_\rho \varphi\,\mathtt{S}\,\psi & \text{iff} & \text{there is a } v \in W \text{ with } vRw \text{ and } \mathcal{F}, v \models_\rho \psi \text{ and} \\
 & & \quad \text{for all } u: \text{ if } vRuRw \text{ then } \mathcal{F}, u \models_\rho \varphi \\[1ex]
\mathcal{F}, w \models_\rho x & \text{iff} & w = \rho(x)
\end{array}$$

$$\begin{aligned}
\mathcal{F}, w \models_\rho @_x\varphi &\text{ iff } \quad \mathcal{F}, \rho(x) \models_\rho \varphi \\
\mathcal{F}, w \models_\rho \downarrow x.\varphi &\text{ iff } \quad \mathcal{F}, w \models_{\rho[x\mapsto w]} \varphi \\
\mathcal{F}, w \models_\rho \exists x.\varphi &\text{ iff } \quad \text{there is } v \in W \text{ with } \mathcal{F}, w \models_{\rho[x\mapsto v]} \varphi
\end{aligned}$$

We write $w \models_\rho \varphi$ instead of $\mathcal{F}, w \models_\rho \varphi$ if the underlying Kripke frame $\mathcal{F}$ can be derived from the context.

The interpretation of Hybrid Logic formulas over Kripke frames $\mathcal{F}$ easily lifts to hybrid Kripke structures $\mathcal{K} = (\mathcal{F}, \rho)$ via $\mathcal{K}, w \models \varphi$ iff $\mathcal{F}, w \models_\rho \varphi$, and then to rooted hybrid Kripke structures $\mathcal{K} = (\mathcal{F}, \rho, w_0)$ via $\mathcal{K} \models \varphi$ iff $\mathcal{F}, w_0 \models_\rho \varphi$.

Two hybrid formulas φ and ψ are equivalent, if they cannot be distinguished by any world of any hybrid Kripke structure, i.e. if for all rooted hybrid Kripke structures $\mathcal{K}$ we have $\mathcal{K} \models \varphi$ iff $\mathcal{K} \models \psi$. In that case we write $\varphi \equiv \psi$.

Later we will also use the *semantics* or *denotation* of a formula φ w.r.t. a Kripke frame $\mathcal{F} = (W, R)$ and a variable interpretation ρ, defined as

$$[\![\varphi]\!]_\rho^{\mathcal{F}} \;:=\; \{w \mid w \in W \text{ and } \mathcal{F}, w \models_\rho \varphi\}$$

Example 1 Let φ and ψ be two formulas of Hybrid Logic and x a first-order variable s.t. $x \notin \mathit{Var}(\varphi, \psi)$. Then the following equivalence holds, i.e. it is in fact an axiom of Hybrid Logic.

$$\varphi \,\mathtt{U}\, \psi \;\equiv\; \downarrow x.\, \mathtt{F}(\psi \wedge \mathtt{H}(\mathtt{P}\, x \rightarrow \varphi))$$

We will show that the left-hand side implies the right-hand side. The opposite direction is entirely analogous. Let $\mathcal{F} = (W, R)$ be a hybrid Kripke frame with $w \in W$ and suppose $w \models_\rho \varphi \,\mathtt{U}\, \psi$ under some interpretation ρ of the second- and first-order variables in φ and ψ. Then there is a $v \in W$ s.t. wRv, $v \models_\rho \psi$, and for all $u \in wR \cap Rv$ we have $u \models_\rho \varphi$. Let $\rho' := \rho[x \mapsto w]$. Then we have $v \models_\rho \psi$ and, since x does not occur in $\mathit{Var}(\psi)$, equally $v \models_{\rho'} \psi$. Furthermore, we have $v \models_{\rho'} \mathtt{H}(\mathtt{P}\, x \rightarrow \varphi)$ because $x \notin \mathit{Var}(\varphi)$ and every predecessor of v that is a successor of w must satisfy φ. Finally, because of wRv we also have $w \models_\rho \downarrow x.\, \mathtt{F}(\psi \wedge \mathtt{H}(\mathtt{P}\, x \rightarrow \varphi))$.

This shows that the inclusion of the binary temporal operators $\mathtt{U}$ and $\mathtt{S}$ is not necessary since they can be expressed in terms of the unary ones and the binding quantifier. On the other hand, it is well-known from temporal logic that the unary ones are not necessary either in the presence of the binary ones: $\mathtt{F}\,\varphi \equiv \top \,\mathtt{U}\, \varphi$ and $\mathtt{P}\,\varphi \equiv \top \,\mathtt{S}\, \varphi$. Nevertheless we keep all of them in, mainly because we want to present a general framework for local model checking of Hybrid Logic which should easily cater for fragments of the logic presented in this form as well. Note for instance that Hybrid Logic without binary temporal operators for example is a very natural fragment.

3 Model Checking

3.1 Local Model Checking Games for Hybrid Logic

We fix a rooted hybrid Kripke structure $\mathcal{K} = ((W, R), \rho_0, w_0)$, and a formula φ_0 of Hybrid Logic. The *game* $\mathcal{G}(\mathcal{K}, \varphi_0)$ is a directed node-labeled graph with node set

$$\mathcal{C} := W \times ((\mathsf{Var}_2(\varphi_0) \to 2^W) \cup (\mathsf{Var}_1(\varphi_0) \to W)) \times FL(\varphi_0)$$

Hence, each node consists of a world, an interpretation of the variables occurring in φ_0, and a (possibly negated) subformula of φ_0.[2] We usually write such a node, which is also called a *configuration*, as $w \vdash_\rho \psi$. The game is played between two players called $\exists$ and $\forall$, and intuitively, player $\exists$ wants to show in a configuration $w \vdash_\rho \psi$ that indeed $w \models_\rho \psi$ holds while player $\forall$ wants to show that $w \not\models_\rho \psi$ holds. Their aim is to traverse the graph until a configuration is reached in which the $\models$-relation is blatantly true or false.

The edge relation is axiomatised by the *game rules* presented in Fig. 1. They consist of a *current configuration* above the line, a *successor configuration* below the line, and to the right possibly a description of the choices to be performed in the current configuration in order to obtain a successor. For instance, in the rule for conjunctions, player $\forall$ chooses an $i \in \{1, 2\}$ since only these two values make sense in the context of the current and successor configuration. The rules for the binary temporal operators are to be read as follows. In a configuration of the form $w \vdash_\rho \varphi \,\mathtt{U}\, \psi$, first player $\exists$ chooses a successor v of w. Then player $\forall$ has the choice to either continue with the configuration $v \vdash_\rho \psi$, or he selects a u which is a successor of w and a predecessor of v and continues with $u \vdash_\rho \varphi$. In case there is no such u he has no choice but to continue with $v \vdash_\rho \psi$.

Starting in the *initial configuration* $w_0 \vdash_{\rho_0} \varphi_0$, the players subsequently perform choices and, thus, create a sequence of configurations called a *play*. Each play is assigned a *winner*, i.e. either of the players. Player $\exists$ wins a play $C_0, C_1, \ldots$ if there is an $n \in \mathbb{N}$, a $w \in W$, and a ρ such that

($\exists$1) $C_n = w \vdash_\rho \top$; or
($\exists$2) $C_n = w \vdash_\rho X$ for some $X \in \mathsf{Var}_2$ and $w \in \rho(X)$; or
($\exists$3) $C_n = w \vdash_\rho \neg X$ for some $X \in \mathsf{Var}_2$ and $w \notin \rho(X)$; or
($\exists$4) $C_n = w \vdash_\rho x$ for some $x \in \mathsf{Var}_1$ and $w = \rho(x)$; or
($\exists$5) $C_n = w \vdash_\rho \neg x$ for some $x \in \mathsf{Var}_1$ and $w \neq \rho(x)$; or
($\exists$6) $C_n = w \vdash_\rho \neg\, \mathtt{F}\, \psi$ for some ψ and $wR = \emptyset$; or
($\exists$7) $C_n = w \vdash_\rho \neg\, \mathtt{P}\, \psi$ for some ψ and $Rw = \emptyset$; or

[2] The advertent reader may later spot a little mismatch between the definition of the configurations and the way that games are considered abstractly later. In abstract games we will assume that in each configuration at most one of the players makes a choice in order to reach a successor configuration. However, according to the game rules we define, there are configurations which require alternating choices between the players to reach a successor. This mismatch can of course be repaired by introducing auxiliary configurations such that first one of the players makes a choice to reach an auxiliary configuration, then the other player makes a choice to reach a proper configuration again. Since this only creates notational overhead without posing any real problems we decide to live with this little mismatch.

$$\frac{w \vdash_\rho \varphi_1 \wedge \varphi_2}{w \vdash_\rho \varphi_i}\ \forall i \qquad \frac{w \vdash_\rho \neg(\varphi_1 \wedge \varphi_2)}{w \vdash_\rho \neg\varphi_i}\ \exists i \qquad \frac{w \vdash_\rho \neg\neg\varphi}{w \vdash_\rho \varphi}$$

$$\frac{w \vdash_\rho \mathtt{F}\,\varphi}{v \vdash_\rho \varphi}\ \exists v \in wR \qquad \frac{w \vdash_\rho \neg\,\mathtt{F}\,\varphi}{v \vdash_\rho \neg\varphi}\ \forall v \in wR \qquad \frac{w \vdash_\rho \mathtt{P}\,\varphi}{v \vdash_\rho \varphi}\ \exists v \in Rw$$

$$\frac{w \vdash_\rho \neg\,\mathtt{P}\,\varphi}{v \vdash_\rho \neg\varphi}\ \forall v \in Rw \qquad \frac{w \vdash_\rho \exists x.\varphi}{w \vdash_{\rho[x \mapsto v]} \varphi}\ \exists v \in W \qquad \frac{w \vdash_\rho \neg\exists x.\varphi}{w \vdash_{\rho[x \mapsto v]} \neg\varphi}\ \forall v \in W$$

$$\frac{w \vdash_\rho \varphi\,\mathtt{U}\,\psi}{v \vdash_\rho \psi \quad |_\forall \quad u \vdash_\rho \varphi}\ \exists v \in wR.\forall u \in wR \cap Rv \qquad \frac{w \vdash_\rho @_x\varphi}{\rho(x) \vdash_\rho \varphi}$$

$$\frac{w \vdash_\rho \neg(\varphi\,\mathtt{U}\,\psi)}{v \vdash_\rho \neg\psi \quad |_\exists \quad u \vdash_\rho \neg\varphi}\ \forall v \in wR.\exists u \in wR \cap Rv \qquad \frac{w \vdash_\rho \neg @_x\varphi}{\rho(x) \vdash_\rho \neg\varphi}$$

$$\frac{w \vdash_\rho \varphi\,\mathtt{S}\,\psi}{v \vdash_\rho \psi \quad |_\forall \quad u \vdash_\rho \varphi}\ \exists v \in Rw.\forall u \in Rw \cap vR \qquad \frac{w \vdash_\rho \downarrow x.\varphi}{w \vdash_{\rho[x \mapsto w]} \varphi}$$

$$\frac{w \vdash_\rho \neg(\varphi\,\mathtt{S}\,\psi)}{v \vdash_\rho \neg\psi \quad |_\exists \quad u \vdash_\rho \neg\varphi}\ \forall v \in Rw.\exists u \in Rw \cap vR \qquad \frac{w \vdash_\rho \neg\downarrow x.\varphi}{w \vdash_{\rho[x \mapsto w]} \neg\varphi}$$

Fig. 1 The model checking game rules for Hybrid Logic

(∃8) $C_n = w \vdash_\rho \neg(\varphi \,\mathtt{U}\, \psi)$ for some φ, ψ and $wR = \emptyset$; or
(∃9) $C_n = w \vdash_\rho \neg(\varphi \,\mathtt{S}\, \psi)$ for some φ, ψ and $Rw = \emptyset$.

Player ∀ wins this play if there is an $n \in \mathbb{N}$, a $w \in W$, and an ρ such that

(∀1) $C_n = w \vdash_\rho \neg\top$; or
(∀2) $C_n = w \vdash_\rho X$ for some $X \in \mathsf{Var}_2$ and $w \notin \rho(X)$; or
(∀3) $C_n = w \vdash_\rho \neg X$ for some $X \in \mathsf{Var}_2$ and $w \in \rho(X)$; or
(∀4) $C_n = w \vdash_\rho x$ for some $x \in \mathsf{Var}_1$ and $w \neq \rho(x)$; or
(∀5) $C_n = w \vdash_\rho \neg x$ for some $x \in \mathsf{Var}_1$ and $w = \rho(x)$; or
(∀6) $C_n = w \vdash_\rho \mathtt{F}\,\psi$ for some ψ and $wR = \emptyset$; or
(∀7) $C_n = w \vdash_\rho \mathtt{P}\,\psi$ for some ψ and $Rw = \emptyset$; or
(∀8) $C_n = w \vdash_\rho \varphi \,\mathtt{U}\, \psi$ for some φ, ψ and $wR = \emptyset$; or
(∀9) $C_n = w \vdash_\rho \varphi \,\mathtt{S}\, \psi$ for some φ, ψ and $Rw = \emptyset$.

We say that player p *wins the game* $\mathcal{G}(\mathcal{K}, \varphi_0)$ iff player p has a winning strategy for this game. A *strategy* is a function $\sigma : \mathcal{C} \to \mathcal{C}$ mapping configurations to successors.[3] A play $\pi = C_0, C_1, \ldots$ *conforms* to strategy σ for player p if, whenever player p makes a choice in configuration C_i for any i, then $C_{i+1} = \sigma(C_i)$. In other words, player p's choices are in accordance with the strategy function. A strategy σ for player p is then called a *winning strategy* if player p wins every play that conforms to σ.

[3] In fact, this is called a *positional strategy* since it determines successor configurations based on the current configuration in a play only, without considering previously visited ones, the so-called *history*. It is well-known that positional strategies, aka history-free ones suffice for the games considered here. Therefore we do not bother to introduce strategies in the general form only to restrict them afterwards anyway.

I.e. such a strategy guides player p's choices to victory regardless of the opponent's choices.

We will now prove that these games correctly characterise the model checking problem for Hybrid Logic meaning that player $\exists$ has a winning strategy exactly for those games on a structure and a formula between which the satisfaction relation holds. One of the advantages of such game-theoretic frameworks is duality: the rules are symmetric in the sense that for every rule there is another with the corresponding (un-)negated formula, and the choices to be made in these rules are the same with the players swapping roles. This makes correctness proofs particularly easy: soundness and correctness are dual to each other in the sense that one proof can be carried out by swapping the players' names and the formulas constructs in the other. However, a few technicalities are needed in advance.

Lemma 1 *Every play of $\mathcal{G}(\mathcal{K}, \varphi)$ has a unique winner.*

Proof All the game rules reduce the size of the formula in the current configuration. This precludes infinite plays. Since there is a rule for every operator in the syntax and their negations, there are only two ways in which a play can stop: either it reaches a configuration $w \vdash_\rho \psi$ such that no rule applies to ψ, or it reaches a configuration in which one of the players cannot make a choice that would be prescribed by the unique rule that applies to this current configuration.

In the former case, ψ must be $\top$, $\neg\top$, an x or $\neg x$ for some $x \in \mathsf{Var}_1$, or X or $\neg X$ for some $X \in \mathsf{Var}_2$. But then winning conditions ($\exists$1)–($\exists$5) and ($\forall$1)–($\forall$5) uniquely determine the winner of the play.

For the other case note that a choice is always possible if the formula in the current configuration is a conjunction or a negation thereof. Also, no play can get stuck with the rules for the $@_x$-operator, the $\downarrow$-binder and the existential quantifiers, resp. their negations. Hence, if a player cannot perform a choice then it must be because the current formula is a modal or temporal operator. In these case, winning conditions ($\exists$6)–($\exists$9) and ($\forall$6)–($\forall$9) determine the play's winner uniquely. $\square$

Both soundness and completeness of the games can be proved by constructing winning strategies for the corresponding players. That will rely on the notion of truth and falsity for a configuration $w \vdash_\rho \psi$ which we will call *true* if $\mathcal{F}, w \models_\rho \psi$ for the underlying Kripke frame $\mathcal{F}$, and *false* otherwise. The next lemma shows that player $\exists$ can preserve truth in a play, and player $\forall$ even must preserve it.

Lemma 2 *Let $\mathcal{G}$ be a Hybrid Logic model checking game and C a configuration that is true and requires player p to make a choice.*

(a) *If $p = \exists$ then there is a successor configuration C' that is true.*
(b) *If $p = \forall$ then any successor configuration C' will be true.*

Proof By case distinction on C. If $C = w \vdash_\rho \mathrm{F}\,\varphi$ and $w \models_\rho \mathrm{F}\,\varphi$ then, according to the semantics of Hybrid Logic, there is a $v \in wR$ such that $v \models_\rho \varphi$. Hence, player $\exists$ can choose this v and continue with the true configuration $v \vdash_\rho \varphi$. The cases of the current formula being $\mathrm{P}\,\varphi$ or $\exists x.\varphi$ are similar.

Dually, player $\forall$ is requested to make a choice in $C = w \vdash_\rho \neg\,\mathrm{F}\,\varphi$ for instance. If we assume it to be true, then we have $v \not\models_\rho \varphi$ for all $v \in wR$ or, equivalently,

$v \models_\rho \neg\varphi$. Hence, no matter which v player $\forall$ chooses, the successor configuration will be true as well. The cases of the current formula being $\neg \mathrm{P}\, \varphi$ or $\neg\exists x.\varphi$ are similar.

Finally, we consider the cases of the temporal operators separately because they require combined subsequent choices by both players. We exemplarily pick out the case of $C = w \vdash_\rho \varphi \mathrm{U} \psi$ for some φ, ψ. If it is true then there must be a $v \in wR$ such that $v \vdash_\rho \psi$ is true as well. Hence, player $\exists$ can choose this v and by continuing with this successor, player $\forall$ preserves truth as well. Moreover, we have $u \models_\rho \varphi$ for any $u \in wR \cap Rv$ by the semantics of Hybrid Logic which shows that all other choices by player $\forall$ will preserve truth too. The cases of the other temporal operators are entirely analogous. □

This is enough to prove completeness of the games. We can then use it and the duality argument mentioned above in order to show soundness as well.

Theorem 1 *Let $\mathcal{K}$ be a rooted hybrid Kripke structure and φ_0 a formula of Hybrid Logic. Player $\exists$ has a winning strategy for the game $\mathcal{G}(\mathcal{K}, \varphi_0)$ iff $\mathcal{K} \models \varphi_0$.*

Proof Let $\mathcal{K} = (\mathcal{F}, \rho_0, w_0)$ with $\mathcal{F} = (W, R)$.

(Completeness) Suppose $\mathcal{F}, w_0 \models_{\rho_0} \varphi_0$. Hence, the initial configuration $w_0 \vdash_{\rho_0} \varphi_0$ is true. According to part (a) of Lemma 2, player $\exists$ can play in such a way that truth is preserved. This defines a strategy for her: in a true configuration choose a successor that is true as well, otherwise choose any. This is in fact a winning strategy because no matter what player $\forall$ does with his choices, every reached configuration must be true according to part (b) of Lemma 2. Finally, note that player $\forall$ can only win a play if it reaches a false configuration since all the configurations named in winning conditions ($\forall$1)–($\forall$7) are false. By Lemma 1, player $\exists$ must win all these plays which shows that preservation of truth is indeed a winning strategy.

(Soundness) Suppose $\mathcal{F}, w_0 \not\models_{\rho_0} \varphi_0$, i.e. the initial configuration is false. It is easy to dualise Lemma 2 to show that player $\forall$ can preserve falsity of configurations and player $\exists$ must preserve falsity. This defines a strategy for player $\forall$ which can be shown to be winning along the lines of the completeness proof: player $\exists$ only wins plays that end in true configurations. Thus, preservation of falsity is a winning strategy for player $\forall$. It should be clear from the definition of winning strategy that it is impossible for two players both to have winning strategies for the same game. Otherwise they could play against each other both using their strategies. The result is a play which would have to be won by both players. But this would contradict Lemma 1. □

3.2 Solving a Hybrid Logic Model Checking Game

The problem of solving a Hybrid Logic model checking game is the following: given a $\mathcal{G} = \mathcal{G}(\mathcal{K}, \varphi)$ decide whether or not player $\exists$ has a winning strategy for $\mathcal{G}$. By Theorem 1 we know that this is equivalent to solving the local model checking problem, namely to deciding whether or not $\mathcal{K} \models \varphi$ holds.

The proof of Lemma 1 shows that every play of $\mathcal{G}(\mathcal{K}, \varphi)$ is finite. Since $\mathcal{G}(\mathcal{K}, \varphi)$ is a reachbility game meaning that winning solely depends on the ability to reach certain configurations, these games can easily be solved using dynamic programming. However, dynamic programming is usually implemented in a bottom-up fashion.

```
Solve(C)  =  if $C \in win_\exists$ then return $\exists$
             if $C \in win_\forall$ then return $\forall$
             let $succs :=$ all successors of $C$ in
             let $p :=$ player who makes a choice in $C$ in
             for all $C' \in succs$ do
                 if Solve($C'$) = $p$ then
                     $win_p := win_p \cup \{C\}$
                     return $p$
             done
             let $o :=$ player $p$'s opponent in
             $win_o := win_o \cup \{C\}$
             return $o$
```

Fig. 2 A top-down dynamic programming model checking algorithm

A top-down approach is needed in order to act locally. An algorithm is sketched in Fig. 2. It is recursive and maintains in two global variables $win_\exists$ and $win_\forall$ sets of configurations that have been visited already. Initially, these sets are empty. The algorithm is called with the game's initial configuration C_0 as the argument. It returns the player who has a winning strategy for the game. In order to simplify the presentation we introduce the convention that, nominally, player p makes a choice in a configuration that makes his/her opponent win according to one of the winning conditions. For example, if $C = v \vdash_\rho \top$ then player $\exists$ wins any play reaching that C, and we say that player $\forall$ has to make a choice in C. Thus, the players win because their opponents cannot make a choice anymore—remember that C has no successors.

Correctness of algorithm `Solve` is not difficult to prove. One can show by induction on the recursion depth that player p wins the game $\mathcal{G}$ starting in position C iff `Solve`$(C) = p$. The more interesting question concerns the algorithm's complexity.

Theorem 2 *The local model checking problem for a finite rooted hybrid Kripke structure $\mathcal{K} = (\mathcal{F}, \rho, w)$ with $\mathcal{F} = (W, R)$ and a formula φ of Hybrid Logic can be solved in time*

$$\mathcal{O}\Big(|W|^{2+bv(\varphi)} \cdot |\varphi| \cdot \log(|W|^{1+bv(\varphi)} \cdot |\varphi|)\Big)$$

and space $\mathcal{O}(|W|^{bv(\varphi)+1} \cdot |\varphi|)$.

Proof First we estimate the number of possible different configurations in $\mathcal{G}(\mathcal{K}, \varphi)$ for some $w_0 \in W$. Remember that a configuration consists of a world $w \in W$, a formula $\psi \in FL(\varphi)$, and an interpretation η of the variables in φ. Note that only the game rules for (possibly negated) formulas of the form $\exists x.\psi$ or $\downarrow x.\psi$ change the interpretation in a configuration. Hence, we have $\eta(y) = \rho(y)$ for every first-order variable x that is not bound in φ, and $\eta(X) = \rho(X)$ for every second-order variable anyway. This provides an upper bound of $|W| \cdot |W|^{bv(\varphi)} \cdot |\varphi|$ on the number of configurations.

Assuming that sets of configurations are stored as balanced trees, the addition and look-up operations can be carried out in time logarithmic in the maximal size of the set, i.e. the number of all configurations.

Lemma 1 implicitly shows that the game graph has no cycles, i.e. the call of `Solve` on some C will terminate before another recursive call of `Solve` can visit C. Hence, whenever `Solve` is called on a configuration C, upon return of this call, C will be added to either $win_\exists$ or $win_\forall$. Thus, a second call of `Solve` on that particular C will terminate in lines 1 or 2 already.

The number of visits to a configuration is then bounded by the number of edges of the game graph. Due to the existential quantifier, each configuration can have up to $|W|$ many successors. Hence, the number of edges is bounded by $|W|^2 \cdot |W|^{bv(\varphi)} \cdot |FL(\varphi)|$. Putting these together we obtain a running time of $\mathcal{O}(|W|^{2+bv(\varphi)} \cdot |\varphi| \cdot \log(|W|^{1+bv(\varphi)} \cdot |\varphi|))$.

The space needed is linear in the number of configurations: the only data structures that need to be maintained are the two sets $win_\exists$ and $win_\forall$. They can be represented in space $\mathcal{O}(|W|^{bv(\varphi)+1} \cdot |\varphi|)$. □

We remark that, by renaming bound variables accordingly, it is always possible to reduce the number of bound variables to the nesting depth of the binders $\exists$ and $\downarrow$ in a formula.

4 Hybrid Logic with Implicit Transitivity Operators

4.1 Local Model Checking and R^+-Frames

Sometimes Hybrid Logic incorporates operators analogous to $\mathtt{F}$, $\mathtt{P}$, $\mathtt{U}$ and $\mathtt{S}$ that are interpreted over R^+, the transitive closure of the accessibility relation R of a hybrid Kripke frame. It is the least (w.r.t. $\subseteq$) set that satisfies $R \cup R \circ R^+ \subseteq R^+$.

One approach to solving the model checking problem for Hybrid Logic enhanced with such operators is to reduce it to the model checking problem for Hybrid Logic as defined above. This is particularly simple: let $\mathcal{F} = (W, R)$ be a Kripke frame, and define $\mathcal{F}^+ := (W, R^+)$. Clearly, $\mathcal{F}^+$ is still a Kripke frame, and Theorem 1 provides a method for solving the model checking problem: first compute R^+ then solve the game for the rooted hybrid Kripke structure whose underlying frame is $\mathcal{F}^+$. However, this defeats locality because in order to compute R^+ one has to know R, i.e. in general traverse the entire frame before the model checking problem is solved.

A first step around this is to compute R^+ itself locally, i.e. whenever the model checking game examines a word w and the game rules for instance require the examination of a successor (in the transitive closure now!) node v then one simply checks the set wR^+ successively instead of wR only. However, this introduces locality superficially. There are issues to consider about the storing of the new relation R^+, and one has to ensure that configurations are only visited at most once in the solving of a game in order to prevent an exponential blow-up.

It is possible to integrate the local computation of the transitive closure into the games directly whilst overcoming such issues and—as we will see below—without affecting the complexity of solving these games to the worse. The idea is the following: in a configuration of the form $w \vdash_\rho \mathtt{F}\,\varphi$ player $\exists$ can still decide to prove φ in an R-successor of w. However, she also receives the choice to postpone this since the successor satisfying φ may actually be an $(R^+ \setminus R)$-successor. From a game-theoretic point of view, these games become more interesting. They are not simple reachability games anymore since indefinite postponing in such choices may lead to circular or in general infinite plays. Hence, the winning conditions need to be amended in order to capture these effects correctly as well. First of all, though, we will introduce corresponding operators for the modalities interpreted over R^+ instead of R in order to be able to distinguish these operators from the normal ones in the games.

4.2 Syntax and Semantics of Hybrid Logic with Transitive Closure Operators

In addition to the syntax defined in Sect. 2.2 we introduce the following operators into Hybrid Logic over the same countably infinite sets of second- and first-order variables Var_2 and Var_1.

$$\varphi \;:=\; \ldots \mid \mathtt{F}^+ \varphi \mid \mathtt{P}^+ \varphi \mid \varphi\, \mathtt{U}^+\, \varphi \mid \varphi\, \mathtt{S}^+\, \varphi$$

As above, we may sometimes use the macros $\mathtt{G}^+ \varphi := \neg\, \mathtt{F}^+ \neg\varphi$ and $\mathtt{H}^+ \varphi := \neg\, \mathtt{P}^+ \neg\varphi$.

In order to incorporate these constructs into the model checking games we need to extend the definition of the Fischer-Ladner closure by clauses for the new constructs. We require, in addition to the earlier closure properties, that

- if $\mathtt{F}^+ \psi \in FL(\varphi)$ then $\mathtt{F}(\psi \vee \mathtt{F}^+ \psi) \in FL(\varphi)$,
- if $\neg\, \mathtt{F}^+ \psi \in FL(\varphi)$ then $\neg\, \mathtt{F}(\psi \vee \mathtt{F}^+ \psi) \in FL(\varphi)$,
- if $\mathtt{P}^+ \psi \in FL(\varphi)$ then $\mathtt{P}(\psi \vee \mathtt{P}^+ \psi) \in FL(\varphi)$,
- if $\neg\, \mathtt{P}^+ \psi \in FL(\varphi)$ then $\neg\, \mathtt{P}(\psi \vee \mathtt{P}^+ \psi) \in FL(\varphi)$,
- if $\psi_1\, \mathtt{U}^+\, \psi_2 \in FL(\varphi)$ then $\downarrow x.\, \mathtt{F}^+(\psi_2 \wedge \mathtt{H}^+(\mathtt{P}^+ x \to \psi_1)) \in FL(\varphi)$ for some $x \notin \mathsf{Var}_1(\psi_1, \psi_2)$,
- if $\neg(\psi_1\, \mathtt{U}^+\, \psi_2) \in FL(\varphi)$ then $\neg(\downarrow x.\, \mathtt{F}^+(\psi_2 \wedge \mathtt{H}^+(\mathtt{P}^+ x \to \psi_1))) \in FL(\varphi)$ for some $x \notin \mathsf{Var}_1(\psi_1, \psi_2)$,
- if $\psi_1\, \mathtt{S}^+\, \psi_2 \in FL(\varphi)$ then $\downarrow x.\, \mathtt{P}^+(\psi_2 \wedge \mathtt{G}^+(\mathtt{F}^+ x \to \psi_1)) \in FL(\varphi)$ for some $x \notin \mathsf{Var}_1(\psi_1, \psi_2)$,
- if $\neg(\psi_1\, \mathtt{S}^+\, \psi_2) \in FL(\varphi)$ then $\neg(\downarrow x.\, \mathtt{P}^+(\psi_2 \wedge \mathtt{G}^+(\mathtt{F}^+ x \to \psi_1))) \in FL(\varphi)$ for some $x \notin \mathsf{Var}_1(\psi_1, \psi_2)$.

As mentioned above, these operators are simply interpreted in the same way as their non-transitive counterparts but over the transitive closure of the accessibility relation R instead of R itself. In addition to the clauses given in Sect. 2.3 we have the following. Let $\mathcal{F} = (W, R)$ again be a Kripke frame, $w \in W$, and ρ an interpretation of the second- and first-order variables with finite domain only.

$$\mathcal{F}, w \models_\rho \mathtt{F}^+ \varphi \quad \text{iff} \quad \text{there is } v \in W \text{ with } wR^+v \text{ and } \mathcal{F}, v \models_\rho \varphi$$

$$\mathcal{F}, w \models_\rho \mathtt{P}^+ \varphi \quad \text{iff} \quad \text{there is } v \in W \text{ with } vR^+w \text{ and } \mathcal{F}, v \models_\rho \varphi$$

$$\begin{aligned}\mathcal{F}, w \models_\rho \varphi\, \mathtt{U}^+\, \psi \quad \text{iff} \quad & \text{there is } v \in W \text{ with } wR^+v \text{ and } \mathcal{F}, v \models_\rho \psi \text{ and} \\ & \text{for all } u:\ \text{if } wR^+uR^+v \text{ then } \mathcal{F}, u \models_\rho \varphi\end{aligned}$$

$$\frac{w \vdash_\rho \mathtt{F}^+ \varphi}{w \vdash_\rho \mathtt{F}(\varphi \vee \mathtt{F}^+ \varphi)} \qquad \frac{w \vdash_\rho \neg\, \mathtt{F}^+ \varphi}{w \vdash_\rho \neg\, \mathtt{F}(\varphi \vee \mathtt{F}^+ \varphi)} \qquad \frac{w \vdash_\rho \mathtt{P}^+ \varphi}{w \vdash_\rho \mathtt{P}(\varphi \vee \mathtt{P}^+ \varphi)} \qquad \frac{w \vdash_\rho \neg\, \mathtt{P}^+ \varphi}{w \vdash_\rho \neg\, \mathtt{P}(\varphi \vee \mathtt{P}^+ \varphi)}$$

$$\frac{w \vdash_\rho \varphi\, \mathtt{U}^+ \psi}{w \vdash_\rho \downarrow x.\, \mathtt{F}^+(\psi \wedge \mathtt{H}^+(\mathtt{P}^+ x \rightarrow \varphi))} \qquad \frac{w \vdash_\rho \neg(\varphi\, \mathtt{U}^+ \psi)}{w \vdash_\rho \neg \downarrow x.\, \mathtt{F}^+(\psi \wedge \mathtt{H}^+(\mathtt{P}^+ x \rightarrow \varphi))}$$

$$\frac{w \vdash_\rho \varphi\, \mathtt{S}^+ \psi}{w \vdash_\rho \downarrow x.\, \mathtt{P}^+(\psi \wedge \mathtt{G}^+(\mathtt{F}^+ x \rightarrow \varphi))} \qquad \frac{w \vdash_\rho \neg(\varphi\, \mathtt{S}^+ \psi)}{w \vdash_\rho \neg \downarrow x.\, \mathtt{P}^+(\psi \wedge \mathtt{G}^+(\mathtt{F}^+ x \rightarrow \varphi))}$$

x is a uniquely chosen fresh variable with $x \notin \mathit{Var}(\varphi) \cup \mathit{Var}(\psi)$

Fig. 3 Additional game rules for Hybrid Logic over transitive frames

$$\mathcal{F}, w \models_\rho \varphi\, \mathtt{S}^+ \psi \quad \text{iff} \quad \text{there is } v \in W \text{ with } vR^+w \text{ and } \mathcal{F}, v \models_\rho \psi \text{ and}$$
$$\text{for all } u:\ \text{if } vR^+uR^+w \text{ then } \mathcal{F}, u \models_\rho \varphi$$

The definitions of formula equivalence, etc., of course remain untouched, and it is also the case that the $\mathtt{F}^+$ and $\mathtt{P}^+$ can be expressed in terms of the $\mathtt{U}^+$ and the $\mathtt{S}^+$ and vice-versa in the presence of the binding quantifier—hence the seemingly strange clauses for the binary operators in the definition of the Fischer-Ladner closure. Again, we keep all operators in the logic as first-class objects in order to avoid translations into full Hybrid Logic should one want to implement model checking games for the natural fragment that contains the unary modal operators only.

4.3 Model Checking Games for Hybrid Logic over Transitive Frames

As with the syntax and the semantics we will simply extend the model checking games defined in Sect. 3.1 by adding new rules and new winning conditions. There are two more rules per new operator, presented in Fig. 3. Note that these rules are all deterministic in the sense that they do not require any player to choose something.

Implicit transitivity requires new winning conditions. Note that formulas of the form $\mathtt{F}^+ \varphi$, and $\mathtt{P}^+ \varphi$ are unfolded to a formula which is larger and contains the original one as a proper subformula. This may lead to cycles in the game graph which represent infinite plays. Note that winning conditions (∃1)–(∃9) and (∀1)–(∀9) only capture finite plays. Hence, we need to assign winners to infinite plays as well.

A play $C_0, C_1, \ldots$ is called a *μ-play*, if there is a φ of the form $\mathtt{F}^+ \psi$, or $\mathtt{P}^+ \psi$, and infinitely many i s.t. $C_i = w_i \vdash_{\rho_i} \varphi$ for some w_i and ρ_i. It is called a *ν-play* if there is a φ of the form $\neg\, \mathtt{F}^+ \psi$, or $\neg\, \mathtt{P}^+ \psi$, and infinitely many i s.t. $C_i = w_i \vdash_{\rho_i} \varphi$ for some w_i and ρ_i.

We add to the winning conditions (∃1)–(∃9) and (∀1)–(∀9) above the following two. Player ∃ wins the infinite play $C_0, C_1, \ldots$ of $\mathcal{G}(\mathcal{K}, \varphi_0)$ if

(∃10) it is a ν-play.

Player ∀ wins this play if

(∀10) it is a μ-play.

We proceed along the same lines to prove correctness of these enhanced games. First of all we need to show that the winning conditions are chosen sensibly. Remember that in the following $\mathcal{G}(\mathcal{K}, \varphi)$ etc. always denotes a model checking game for Hybrid Logic including the implicit transitivity operators and, thus, the addtional rules and winning conditions defined above.

Lemma 3 *Every play of $\mathcal{G}(\mathcal{K}, \varphi)$ has a unique winner.*

Proof It is not hard to see that winning conditions (∃1)–(∃9) and (∀1)–(∀9) still determine the winner of a finite play uniquely. Thus, it remains to be seen that every infinite play matches exactly one of (∃10) and (∀10).

Let $C_0, C_1, \ldots$ be an infinite play. Note that all the rules in Fig. 1 strictly decrease the size of the formula in a configuration whereas the rules in Fig. 3 strictly increase them. Furthermore, the rules for the binary operators in Fig. 3 eliminate a binary construct from the formula at hand. Hence, the play must use the upper four rules of Fig. 3 infinitely often. There are only finitely many of them and the Fischer-Ladner closure of any formula is finite. Thus, by the Pigeon Hole Principle, there is a formula of the form $\mathtt{F}^+\,\psi$, $\mathtt{P}^+\,\psi$, $\neg\,\mathtt{F}^+\,\psi$, or $\neg\,\mathtt{P}^+\,\psi$ that occurs infinitely often in this play. Hence, any infinite play is at least a μ-play or a ν-play and therefore captured by at least one of the winning conditions (∃10) and (∀10). It remains to be seen that at most one of them can apply.

Suppose there are ψ and ψ' of that form such that $C_i = w_i \vdash_{\rho_i} \psi$ for infinitely many i and $C_j = w_j \vdash_{\rho_j} \psi'$ for infinitely many j. Then, for every such C_i there is such a C_j with $j > i$ and vice-versa. Now take a configuration with formula χ and a successor configuration with formula χ'. Then $\chi' \in FL(\chi)$. Hence, in this case we would have both $\psi' \in FL(\psi)$ and $\psi \in FL(\psi')$. But this is only possible if $\psi = \psi'$ which shows an even stronger statement than needed: the formula that determines whether the play is either a μ-play or a ν-play is even unique. □

The notions of truth and falsity of a configuration remain the same. Note that they do not appeal to (in-)transitivity of frames, but merely to the semantics of Hybrid Logic which is still well-defined in the presence of the implicit transitivity operators. The next step is to prove preservation of truth, resp. falsity, in that presence. This simply extends Lemma 2 to the richer logic.

Lemma 4 *Let $\mathcal{G}$ be a Hybrid Logic model checking game and C a configuration that is true and requires player p to make a choice.*

(a) *If $p = \exists$ then there is a successor configuration C' that is true.*
(b) *If $p = \forall$ then any successor configuration C' will be true.*

Proof All that remains to be done is to consider the additional game rules. They do not impose any choices on the players but are deterministic, i.e. there is always exactly one successor configuration. Since they do not change the world w nor the variable interpretation ρ in the transition from the current to the successor configuration the claim boils down to showing that the formulas in the current and the successor configuration are equivalent. This, however, is the case for all rules which is easily checked.

The upper four rules implement the well-known unfolding equivalences of the temporal operators, and the equivalence preservation of the other four rules is shown in the same way as is done in Ex. 1. □

Finally, we show correctness of the enhanced games by extending the proof of Theorem 1. It is tempting to define player $\exists$'s strategy again as "preserve truth" like it is done in the proof of Theorem 1. However, this is not necessarily a winning strategy anymore which is due to the fact that the games are not generally acyclic. Take for example the Kripke frame $\mathcal{F} = (\{w\}, R)$ with $R = \{(w, w)\}$ and any variable interpretation ρ which we will drop in $\models_\rho$ since it is irrelevant. Consider the formula $\mathtt{F}^+ \top$. Clearly, we have $w \models \mathtt{F}^+ \top$ and therefore, by equivalence, $w \models \mathtt{F}(\top \vee \mathtt{F}^+ \top)$. This is given because wRw and $w \models \top$ as well as $w \models \mathtt{F}^+ \top$ which was already stated above. This means that in the configuration $w \vdash_\rho \top \vee \mathtt{F}^+ \top$ both options for player $\exists$ allow her to preserve truth. However, it is not ok to just take any of them. This was sufficient for the acyclic reachability games for Hybrid Logic without implicit transitivity. There, if two successor configurations are true, then they are both true on their own account. Here, because of possible cycles in the game graph, one successor may only be true because another successor is true. Note that this is exactly what is happening in this example: we have $w \models \mathtt{F}^+ \top$ only because wRw and $w \models \top$. Thus, the strategy must take a closer look at the formulas in a configuration.

Unfortunately, this is not the only situation which disqualifies the simple truth-preservation strategy. Take the hybrid Kripke structure $\mathcal{K} = ((\{w, v, u\}, R), \rho)$ with $R = \{(w, w), (w, v), (v, u)\}$ and $\rho(X) = \{u\}$. Then we have $\mathcal{K}, x \models \mathtt{F}^+ X$ for any $x \in \{w, v\}$. Hence, the configuration $w \vdash_\rho \mathtt{F}(X \vee \mathtt{F}^+ X)$ is true, and player $\exists$ has to choose an $x \in \{w, v\}$ to continue with $x \vdash_\rho X \vee \mathtt{F}^+ X$. In any case, the disjunct X will be false in x, therefore we may assume that she gets to $x \vdash_\rho \mathtt{F}^+ X$ afterwards. If she selected $x = w$ then she makes the play cyclic and it is won by player $\forall$ with his winning condition ($\forall$7). Thus, the strategy must also take a closer look at the worlds in a configuration.

Fortunately, there is a very simple syntactic criterion which we can incorporate into player $\exists$'s strategy and which deals with the first problem. The second one can be conquered with a little definition. Let $\mathcal{F} = (W, R)$ be a Kripke frame and $w, v \in W$. The *R-distance* between w and v is the length of a shortest non-empty R-path from w to v.

$$\delta_R(w, v) \;:=\; \begin{cases} \min\{i \mid wR^i v\}, & \text{if } wR^+v \\ \infty, & \text{otherwise} \end{cases}$$

where $R^1 := R$, $R^{i+1} := R \circ R^i$ as usual. Note that $\delta_R(w, v) \geq 1$ for all w, v, and $\delta_R(w, w) = \infty$ is possible.

Theorem 3 *Let $\mathcal{K}$ be a rooted hybrid Kripke structure and φ_0 a formula of Hybrid Logic. Player $\exists$ has a winning strategy for the game $\mathcal{G}(\mathcal{K}, \varphi_0)$ iff $\mathcal{K} \models \varphi_0$.*

Proof Let $\mathcal{K} = (\mathcal{F}, \rho_0, w_0)$ with $\mathcal{F} = (W, R)$.

(Completeness) Suppose $\mathcal{K} \models \varphi_0$. We need to construct a strategy for player $\exists$. This is defined as follows. Suppose player $\exists$ has to make a choice in a configuration C.

If C is not true, then she can choose any successor. If C is true then let $s(C) = \{C' \mid C'$ is a true successor of $C\}$. We need to make a case distinction on the type of C.

- If $C = w \vdash_\rho \mathtt{F}(\psi \vee \mathtt{F}^+ \psi)$ then, because of truth, the set $U := \{u \mid wR^+u$ and $u \models_\rho \psi\}$ is non-empty. Hence, there is a (not necessarily unique) $u_0 \in U$ s.t. $\delta_R(w, u)$ is minimal among all $u \in U$. Let $i := \delta_R(w, u_0)$. Then $1 \le i < \infty$, $wR^i u_0$, and $wR^j u_0$ is not the case for all $1 \le j < i$. Since $1 \le i$ there must be a (not necessarily unique, even if u_0 is unique) v such that wRv and $\delta_R(v, u_0) = i-1$. Now player $\exists$ chooses the successor with world v.
- If $C = w \vdash_\rho \mathtt{P}(\psi \vee \mathtt{P}^+ \psi)$ then player $\exists$ does the same with $U = \{u \mid uR^+w$ and $u \models_\rho \psi\}$, u_0 chosen such that $\delta_R(u_0, w)$ is minimal, and v a predecessor of w on a shortest R-path from u_0 to w.
- In all other cases she chooses a $C' \in s(C)$ with a smallest formula component.

Now suppose $\mathcal{F}, w_0 \models_{\rho_0} \varphi_0$. Thus the initial configuration C of $\mathcal{G}(\mathcal{K}, \varphi_0)$ is true. Note that we always have $|s(C)| \ge 1$ because a true configuration must have at least one true successor unless it offers a choice for player $\forall$.

We need to show that this strategy of preserving truth whilst minimising formula sizes and R-distances is indeed a winning strategy. Note that this is a further restriction compared to the strategy of simple truth-preservation defined in the proof of Theorem 1. There we already showed that player $\forall$ cannot win any finite play against that strategy. Hence, he cannot win any finite play against the strategy defined here either.

Now take an infinite play $C_0, C_1, \ldots$ and assume that it is won by player $\forall$, i.e. winning condition ($\forall$10) applies. We only consider the case of the responsible formula being $\mathtt{F}^+ \psi$ here, the other with $\mathtt{P}^+ \psi$ is dealt with in exactly the same way because of the symmetry between $\mathtt{F}^+$ and $\mathtt{P}^+$. Remember that all C_i are true, and that there are infinitely many i_j with $C_{i_j} = w_j \vdash_{\rho_j} \mathtt{F}^+ \psi$ for some w_j and ρ_j. Then we have $C_{i_j+1} = w_i \vdash_{\rho_i} \mathtt{F}(\psi \vee \mathtt{F}^+ \psi)$ for all such i. A close inspection of the rules shows that we have $w_j R^+ w_{j+1}$ for all $j \in \mathbb{N}$ because the rule for $\mathtt{F}$ requires player $\exists$ to choose a successor of the current world, and the rule for disjunctions leaves the current word unchanged.

Now define the *rank* of a configuration $C_{i_j} = w_j \vdash_{\rho_j} \mathtt{F}(\psi \vee \mathtt{F}^+ \psi)$ as $r(C_{i_j}) = \min\{\delta(w_j, v) \mid w_j R^+ v, v \models_{\rho_j} \psi\}$. Because of truth of all C_{i_j} we have $1 \le r(C_{i_j}) < \infty$ for all $j \in \mathbb{N}$. Furthermore, because of $w_j R^+ w_{j+1}$ for all $j \in \mathbb{N}$ we also have $r(C_{i_{j+1}}) < r(C_{i_j})$. This, however, is a contradiction. We conclude that the assumption was wrong, i.e. that the play cannot have been a μ-play after all. According to Lemma 3 it must have been won by player $\exists$ instead which shows that the strategy defined above is indeed a winning strategy.

(Soundness) Analogous to the soundness part in the proof of Theorem 1, i.e. by duality from the completeness proof. □

4.4 Deciding the Model Checking Problem with Implicit Transitivity

Due to the presence of operators with implicit transitivity the model checking games are neither acyclic nor simple reachability games anymore. It is therefore not surprising to hear that the dynamic programming algorithm `Solve` from Fig. 2 would not solve such games correctly. Instead we need an algorithm that can cope with cycles in the game graph. But first we take a closer look at the type of game that is created by the implicit transitivity operators.

The model checking games for Hybrid Logic then become so-called *weak Büchi games*. A Büchi game is a special case of a parity game in which every node of the game graph is assigned a natural number called the *priority*. The winner of an infinite play is determined by the parity of the least priority that occurs infinitely often in that play. Usually, the winner is player $\exists$ if this number is even and player $\forall$ otherwise.

In a *Büchi game*, only the numbers 0 and 1 are assigned to a node. A Büchi game is called *weak* if it can be decomposed into strongly connected components such that all nodes in a component have the same priority. Note that this generalises the model checking games of the previous section: in an acyclic game every strongly connected component has exactly one node. Thus, the weakness requirement is trivially satisfied.

Theorem 4 *The local model checking game $\mathcal{G}(\mathcal{K}, \varphi)$ for a rooted hybrid Kripke structure $\mathcal{K} = (\mathcal{F}, \rho, w)$ and a formula φ of Hybrid Logic with implicit transitivity is a weak Büchi game.*

Proof Clearly, the model checking games are games. Hence, all that remains to be shown is how the priorities 0 and 1 can be assigned to the strongly connected components of $\mathcal{G}(\mathcal{K}, \varphi)$ such that the least priority occurring infinitely often in μ-plays is 1, while the least priority occurring infinitely often in ν-plays is 0.

Consider the SCC decomposition of the game graph $\mathcal{G}(\mathcal{K}, \varphi)$. First note that no SCC can contain two different configurations $C = w \vdash_\rho \psi$ and $C' = w' \vdash_{\rho'} \psi'$ such that $\psi = \mathtt{F}\,\chi$ or $\psi = \mathtt{P}\,\chi$, and $\psi' = \neg\,\mathtt{F}\,\chi'$ or $\psi' = \neg\,\mathtt{P}\,\chi'$. If this was the case then one could construct a play in which both occur infinitely often which would contradict Lemma 3. Hence, the following is well-defined: all configurations in an SCC containing a $w \vdash_\rho \mathtt{F}\,\chi$ or $w \vdash_\rho \mathtt{P}\,\chi$ receive priority 1, all others receive priority 0. It is not difficult to see that an infinite play is a μ-play iff it visits configurations with priority 1 infinitely often and configurations with priority 0 only finitely often. $\square$

Solving a weak Büchi game locally is not much more difficult than solving acyclic reachability games. Nevertheless, this is not trivial, and it was only discovered after some years of research on closely related problems. Here we refrain from giving an algorithm that solves such games locally. Instead we point out the relevant literature.

The interest with solving weak Büchi games and efficient algorithms for solving them arose with the discovery of the importance of temporal logic model checking for program verification. One of the earliest temporal logics used there was CTL. The model checking problem for CTL easily reduces to the problem of solving weak Büchi games. In fact, solving such games is equivalent under linear-time reductions to the model checking problem for the alternation-free μ-calculus. Several algorithms have been invented for that problem and thus also for solving weak Büchi games. Global algorithms can be found in Cleaveland and Steffen (1992); Emerson and Lei (1986), and local ones in Bhat and Cleaveland (1996); Andersen (1994). Here we adopt the complexity bounds of the latter. It should be clear, though, that binary operators in the formula increase the complexity because they get eliminated at the cost of the introduction of a new bound first-order variable, and the complexity of solving acyclic reachability games is already exponential in the number of bound variables.

For a formula φ of Hybrid Logic with transitivity operators we define

$$bo(\varphi) := |\{\psi \mid \psi \in FL(\varphi), \text{ and } \psi = \psi_1 \mathtt{U} \psi_2 \text{ or } \psi = \psi_1 \mathtt{S} \psi_2 \text{ or } \psi = \neg(\psi_1 \mathtt{U} \psi_2) \text{ or } \psi = \neg(\psi_1 \mathtt{S} \psi_2)\}|$$

Proposition 1 *The local model checking problem for a hybrid Kripke structure $\mathcal{K} = (\mathcal{F}, \rho)$ with $\mathcal{F} = (W, R)$ and a formula φ of Hybrid Logic with implicit transitivity operators can be solved in time*

$$\mathcal{O}\Big(|W|^{2+bv(\varphi)+bo(\varphi)} \cdot |\varphi| \cdot \log(|W|^{1+bv(\varphi)+bo(\varphi)} \cdot |\varphi|)\Big)$$

and space $\mathcal{O}(|W|^{1+bv(\varphi)+bo(\varphi)} \cdot |\varphi|)$.

Proof Here, the number of configurations is bounded by $|W|^{1+bv(\varphi)+bo(\varphi)} \cdot |\varphi|$ since every binary operator introduces a new bound variable. Because of the rules for existential quantifiers, each configuration can have up to $|W|$ many successors. Hence, the number of edges of the game graph is bounded by $|W|^{2+bv(\varphi)+bo(\varphi)} \cdot |\varphi|$. The algorithm of Andersen (1994) solves weak Büchi games locally in time $\mathcal{O}(m \cdot \log n)$ where m is the number of edges and n the number of nodes in the graph, provided that $m \geq n$ which is the case here. It only requires space for data structures storing the graphs nodes, i.e. $\mathcal{O}(n)$. □

We conclude this section by pointing out there there is also a local model checking algorithm specifically designed for CTL (Vergauwen and Lewi 1993), and that CTL is obviously closely related to Hybrid Logic even though the unfolding of binary temporal operators that is usually done in CTL is not sound for Hybrid Logic. There is also a local algorithm that solves parity games in general and therefore also weak Büchi games (Stevens and Stirling 1998).

5 Improving Global Model Checking

5.1 Exploiting Monotonicity

While the temporal operators of Hybrid Logic are geared towards local model checking techniques, the first-order operators pose more difficulties. Take for instance a formula of the form $\exists x.\varphi$ which is to be evaluated over a hybrid Kripke structure $\mathcal{K} = (\mathcal{F}, \rho)$ with $\mathcal{F} = (W, R)$. The approach to global model checking proposed in Franceschet and de Rijke (2006) suggests to check for every $w, v \in W$ whether $\mathcal{F}, w \models_{\rho[x \mapsto v]} \varphi$. While the search through all such w is eliminated in the local top-down approach of the previous section, it is still necessary to check all v. This is reflected in the fact that a configuration of the form $w \vdash_\rho \exists x.\varphi$ has a successor for every $v \in W$. Similar observations hold for the binder $\downarrow$.

Even though some operators of Hybrid Logic may seem inherently global and not to admit easy localisation of model checking procedures, there are cases in which global model checking procedures can at least be optimised away from such brute-force methods that simply exploit all possible worlds. Here we consider one such case which is given by monotonicity. The crucial trick to the optimised evaluation of first-order constructs is to view first-order variables as special second-order ones or, in a different terminology, to regard nominals as propositions.

Let φ be a formula of Hybrid Logic (possibly with implicit transitivity operators from now on), $x \in \mathsf{Var}_1$, and $X \in \mathsf{Var}_2$. We say that φ is *positive* in x, resp. X if every free occurrence of x, resp. X, is under the scope of an even number of negation symbols.

Lemma 5 *Let $X \in \mathsf{Var}_2$ and φ be a formula of Hybrid Logic that is positive in X. Then for all hybrid Kripke structures $\mathcal{K} = (\mathcal{F}, \rho)$ with $\mathcal{F} = (W, R)$ and all $U, V \subseteq W$ we have: $U \subseteq V$ implies $[\![\varphi]\!]^{\mathcal{F}}_{\rho[X\mapsto U]} \subseteq [\![\varphi]\!]^{\mathcal{F}}_{\rho[X\mapsto V]}$.*

We omit the details of the proof since this lemma is fairly standard. It can be shown by induction on the structure of φ. The lemma's statement is of course too weak for an inductive proof because X may occur under an odd number of negation symbols inside a subformula of φ. One therefore has to prove the stronger statement which, in addition, asserts that the mapping induced by φ is antitone if X only occurs under an odd number of negation symbols.

5.2 Using Greatest Fixpoints to Approximate Binding Quantifiers

Next we extend the syntax of Hybrid Logic by a restricted second-order quantifier.

$$\varphi \ := \ \ldots \mid \nu X.\varphi$$

where $X \in \mathsf{Var}_2$ and φ on the right-hand side is positive in X. This may be seen as an auxiliary definition which will only be used in order to speed up the evaluation of other operators. It can also be incorporated into Hybrid Logic as a first-class object which would increase its expressive power at a moderate increase in computational complexity.

Such a formula $\nu X.\varphi$ is interpreted as the greatest fixpoint of the function that takes a set of worlds V and returns the semantics of φ under the proviso that X is interpreted as V. According to the Knaster-Tarski Theorem Tarski (1955) greatest fixpoints of monotone functions exist in complete lattice (here the powerset of the underlying set of worlds) and equal the union of all post-fixpoints. Hence, we can define the semantics of this operators as follows.

$$[\![\nu X.\varphi]\!]^{\mathcal{F}}_{\rho} \ := \ \bigcup \left\{ V \subseteq W \mid V \subseteq [\![\varphi]\!]^{\mathcal{F}}_{\rho[X\mapsto V]} \right\}$$

or, equivalently,

$$\begin{aligned} \mathcal{F}, w \models_{\rho} \nu X.\varphi \quad \text{iff} \quad & \text{there is a } V \subseteq W \text{ with } w \in V \text{ and} \\ & \text{for all } v \in V : \mathcal{F}, v \models_{\rho[X\mapsto V]} \varphi \end{aligned}$$

Greatest fixpoints are easy to compute on a Kripke frame $\mathcal{F}$ with world set W using the standard approximation technique:

```
Evaluate($\nu X.\varphi, \rho$)   =   if $\varphi$ not positive in $X$ then fail
                                        $V :=$ W
                                        $V' := \emptyset$
                                        while $V' \neq V$ do
                                            $V' := V$
                                            $V :=$ Evaluate($\varphi, \rho[X \mapsto V]$)
                                        return $V$
```

It is standard to show that this procedure terminates after at most $|W|$ many iterations of the `while`-loop due to monotonicity of the semantics of φ in X, and it returns $[\![\nu X.\varphi]\!]^{\mathcal{F}}_{\rho}$.

The following lemma links greatest fixpoint quantifiers to the binding quantifier.

Lemma 6 *Let $\mathcal{F}$ be a Kripke frame, ρ a variable interpretation, and φ a formula that is Positive in x and does not contain X freely. Then we have $[\![\downarrow x.\varphi]\!]^{\mathcal{F}}_{\rho} \subseteq [\![\nu X.\varphi[X/x]]\!]^{\mathcal{F}}_{\rho}$.*

Proof Suppose $w \in [\![\downarrow x.\varphi]\!]^{\mathcal{F}}_{\rho}$, i.e. $\mathcal{F}, w \models_{\rho} \downarrow x.\varphi$. According to the semantics of the binder $\downarrow$, we have $\mathcal{F}, w \models_{\rho[x \mapsto w]} \varphi$. Since every first-order variable can be seen as a second-order variable whose interpretations must be singletons we also have $\mathcal{F}, w \models_{\rho[X \mapsto \{w\}]} \varphi[X/x]$ if X does not occur freely in φ. But then we have $\{w\} \subseteq [\![\varphi[X/x]]\!]^{\mathcal{F}}_{\rho[X \mapsto \{w\}]}$. In other words, $\{w\}$ is a post-fixpoint of the mapping associated with $\varphi[X/x]$. This post-fixpoint is included in the greatest fixpoint, therefore we have $\{w\} \subseteq [\![\nu X.\varphi[X/x]]\!]^{\mathcal{F}}_{\rho}$ which yields $\mathcal{F}, w \models_{\rho} \nu X.\varphi[X/x]$. □

Intuitively, the statement of the lemma is clearly true: $\downarrow x.\varphi$ is satisfied by all worlds that satisfy φ under the assumption that themselves and only themselves are called x. On the other hand, $\nu X.\varphi[X/x]$ is satisfied by all worlds that satisfy φ under the assumption that themselves and possibly other worlds as well are called X. The latter statement is clearly weaker than the former.

The global model checking procedure of Franceschet and de Rijke (2006) works by recursion on the structure of the formula it receives as an input. Hence, it can be seen as a big case distinction on the top-level operator in the formula, and contains clauses for every such possibility. In the clause for the binding quantifier the suggestion is to evaluate a formula of the form $\downarrow x.\varphi$ over a hybrid Kripke structure $\mathcal{K} = (\mathcal{F}, \rho)$ with $\mathcal{F} = (W, R)$ as follows. For every $w \in W$ compute recursively the semantics of φ under the variable interpretation $\rho[x \mapsto w]$. Then $[\![\downarrow x.\varphi]\!]^{\mathcal{F}}_{\rho}$ consists of all such w that are included in $[\![\varphi]\!]^{\mathcal{F}}_{\rho[x \mapsto w]}$. Lemma 6 contains an idea for a possible improvement. Figure 4 presents an alternative clause in a recursive global model checker that deals with formulas of this kind. It is assumed that `Evaluate` is defined for all formulas which are not of this kind.

The difference in comparison to the naïve evaluation of binding quantifiers is that worlds w which are not included in the greatest fixpoint of the associated mapping are not considered because, by Lemma 6, they cannot satisfy $\downarrow x.\varphi$. On the other hand, this requires the pre-computation of $[\![\nu X.\varphi[X/x]]\!]^{\mathcal{F}}_{\rho}$. Thus, one has to find a balance between the additional cost of this pre-computation and the benefit of the reduced

Fig. 4 Evaluation of the binding quantifier with pre-computation

```
Evaluate($\downarrow x.\varphi$,$\rho$)  =
    if $\varphi$ not positive in $x$ then fail
    let $V$ = Evaluate($\nu X.\varphi[X/x]$,$\rho$) in
    $U := \emptyset$
    for all $w \in V$ do
        let $T$ = Evaluate($\varphi$,$\rho[x \mapsto w]$) in
        if $w \in T$ then
            $U := U \cup \{w\}$
    return $U$
```

search space for the binding quantifier. On the other hand, the benefit is only small if the greatest fixpoint is close to the entire set of worlds. Since greatest fixpoints are usually computed in an approximation "from above", the pre-computation should finish quickly in that case.

5.3 Slicing Kripke Frames

We present a second approach to evaluating binding quantifiers over monotone formulas. It does not depend on greatest fixpoints and can be used to speed up the computation of existential quantifiers as well. The idea is—like in the previous case—to consider nominals, i.e. first-order variables, as propositions for a moment. Remember that in both the (global) evaluation of a binding or an existential quantifier over a variable x, one would have to search through all worlds for those that can be called x s.t. a certain property holds. Here we will approximate these worlds from above by, again, replacing the first-order variable x with a second-order variable X which is then narrowed down to singleton sets in a divide-and-conquer fashion.

This approach is particularly prone to symbolically represented Kripke frames because in the division step one has to split a set of worlds into two, preferably of equal size. In symbolic model checking, a set of worlds of cardinality 2^n is represented by a boolean function in n variables, and these functions are stored as BDDs. Slicing a set into two equal halves is simply done by fixing the value of one of these variables to be 0 or 1 respectively.

An evaluation procedure for the binding quantifier based on the slicing approach is presented in Fig. 5. Again, we assume X not to occur freely in φ, and φ to be positive in x. Correctness of this procedure is then a simple consequence of Lemma 5. Suppose $v \in V \subseteq W$. Then $v \notin [\![\varphi[X/x]]\!]^{\mathcal{F}}_{\rho[X \mapsto V]}$ implies $v \notin [\![\varphi]\!]^{\mathcal{F}}_{\rho[x \mapsto v]}$, i.e. $\mathcal{F}, v \not\models_\rho \downarrow x.\varphi$.

The procedure in Fig. 5 simply tests all $v \in W$ for satisfaction of the formula $\downarrow x.\varphi$ but ignores those that do not even satisfy $\varphi[X/x]$ under an interpretation that maps X to a set that they are contained in.

In the worst case when many worlds satisfy $\downarrow x.\varphi$, this procedure is worse than the naïve one because it has to test all worlds for satisfaction, which the naïve one does as well, but in addition also has to evaluate $\varphi[X/x]$ a number of times which is bounded by $|W| - 1$. On the other hand, if only a few worlds satisfy $\downarrow x.\varphi$, say m out of n, then the divide-and-conquer strategy pays off because only $\mathcal{O}(m \cdot \log n)$ many recursive

```
Evaluate($\downarrow x.\varphi$,$\rho$)  =
    if $\varphi$ not positive in $x$ then fail
    Bind-Eval($\varphi$,$\rho$,$W$)

Bind-Eval($\varphi$,$\rho$,$V$)  =
    if $V = \{v\}$ for some $v \in W$ then
        if $v \in$ Evaluate($\varphi$,$\rho[x \mapsto v]$) then
            return $\{v\}$
        else
            return $\emptyset$
    else
        decompose $V$ into $V_0 \cup V_1$
        $U := \emptyset$
        for $i = 0, 1$ do
            if Evaluate($\varphi[X/x]$,$\rho[X \mapsto V_i]$) $\cap V_i \neq \emptyset$ then
                $U := U \cup$ Bind-Eval($\varphi$,$\rho$,$V_i$)
        return $U$
```

Fig. 5 Divide-and-conquer evaluation of the binding quantifier

```
Evaluate($\exists x.\varphi$,$\rho$)  =  Exists-Eval($W$)

Exists-Eval($V$)  =
    if $V = \{v\}$ for some $v \in W$ then
        if Evaluate($\varphi$,$\rho[x \mapsto v]$) $\neq \emptyset$
            return $W$
        else
            return $\emptyset$
    else
        decompose $V$ into $V_0 \cup V_1$
        if Evaluate($\varphi[X/x]$,$\rho[X \mapsto V_0]$) $\neq \emptyset$ then
            if Exists-Eval($\varphi$,$\rho$,$V_0$) $= \emptyset$ then
                return Exists-Eval($V_1$)
            else
                return $W$
        else
            return Exists-Eval($V_1$)
```

Fig. 6 Divide-and-conquer evaluation of the existential quantifier

calls are needed to complete the evaluation. Note that the special case of $m = 1$ would amount to binary search in a Kripke structure.

At last, we show how this technique can be used to accelerate the evaluation of the other notoriously global operator, the existential quantifier in a formula $\exists x.\varphi$ where φ is positive in x. Again, rather than enumerating all worlds and testing them one-by-one, we try to exclude large parts of the set of worlds by slicing. The procedure is presented in Fig. 6. Again, correctness is a simple consequence of Lemma 5. Note that `Exists-Eval` either returns $\emptyset$ or the entire set of worlds W in any recursive call.

6 Conclusion and Further Work

Driven by the motivation of having Hybrid Logic model checking algorithms that can cope with large structures we have developed a game-based framework which reduces the local model checking problem to the problem of solving simple games, acyclic reachability ones or weak Büchi games. Several solvers for such games exist and can be used to solve the model checking problem for Hybrid Logic locally now.

We have also argued that the global model checking approach presented in Franceschet and de Rijke (2006) should not be considered as the final proposal for global model checking of Hybrid Logic by showing how certain features in the logic (here: monotone formulas) may be used to optimise their algorithms.

The last point opens up an obvious line of further work: determine further cases in which the global model checking can be improved by making it more local. It is, for example, thinkable, that antitone formulas allow similar optimisations as monotone ones.

It is also clear that the game-based approach presented here should undergo a rigorous empirical evaluation, i.e. it should be implemented in a model checking tool for Hybrid Logic and tested on a large set of benchmarks in order to assess its practicality and possibly reveal other ideas for optimisations.

References

Andersen, H. R. (1994). Model checking and boolean graphs. *TCS, 126*(1), 3–30.

Areces, C. (2000). *Logic engineering. The case of description and hybrid logics*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands.

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

Areces, C., & ten Cate, B. (2006). Hybrid logics. In P. Blackburn, F. Wolter, & J. van Benthem (Eds.), *Handbook of modal logics*. Elsevier.

Bhat, G., & Cleaveland, R. (1996). Efficient local model-checking for fragments of the modal μ-calculus. In T. Margaria & B. Steffen (Eds.), *Proceedings of the 2nd international workshop on tools and algorithms for construction and analysis of systems, TACAS'96, LNCS* (Vol. 1055, pp. 107–126). Springer.

Bidoit, N., Cerrito. S., & Thion, V. (2004). A first step towards modeling semistructured data in hybrid multimodal logic. *Journal of Applied Non-Classical Logics, 14*(4), 447–475.

Blackburn, P. (1993). Modal logic and attribute value structures. In M. de Rijke (Ed.), *Diamonds and defaults, synthese language library* (pp. 19–65). Dordrecht: Kluwer Academic Publishers.

Blackburn, P., & Tzakova, M. (1998). Hybridizing concept languages. *Annals of Math and AI, 24*(1–4), 23–49.

Bull, R. (1970). An approach to tense logic. *Theoria, 36*, 282–300.

Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., & Hwang, L. J. (1992). Symbolic model checking: 10^{20} states and beyond. *Information and Computation, 98*(2), 142–170.

Cleaveland, R. (1990). Tableau-based model checking in the propositional μ-calculus. *Acta Informatica, 27*(8), 725–748.

Cleaveland, R., & Steffen, B. (1992). A linear-time model-checking algorithm for the alternation-free modal μ-calculus. In *Proceedings of the 3rd international conference on computer aided verification, CAV'91, LNCS* (Vol. 575, pp. 48–58). Springer.

Emerson, E. A., & Clarke, E. M. (1982). Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming, 2*(3), 241–266.

Emerson, E. A., & Lei, C. L. (1986). Efficient model checking in fragments of the propositional μ-calculus. In *Symposium on Logic in Computer Science* (pp. 267–278). IEEE, Washington, D.C., USA.

Franceschet, M., & de Rijke, M. (2003). Model checking for hybrid logics. In *Proceedings of the 3rd international workshop methods for modalities, M4M-3*.

Franceschet, M., & de Rijke, M. (2006). Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic, 4*(3), 279–304.

Goranko, V. (1996). Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language, and Information, 5*(1), 1–24.

Jurdziński, M. (2000). Small progress measures for solving parity games. In H. Reichel & S. Tison (Eds.), *Proceedings of the 17th annual symposium on theoretical aspects of computer science, STACS'00, LNCS* (Vol. 1770, pp. 290–301). Springer.

Kozen, D., & Tiuryn, J. (1990). Logics of programs. In J. van Leeuwen (Ed.), *Handbook of theoretical computer science, vol B: Formal models and semantics, chap 14* (pp. 789–840). New York, USA: Elsevier and MIT Press.

Passy, S., & Tinchev, T. (1991). An essay in combinatory dynamic logic. *Information and Computation, 93*(2), 263–332.

Queille, J. P., & Sifakis, J. (1982). Specification and verification of concurrent systems in CESAR. In *Proceedings of the 5th symposium on programming, LNCS* (Vol. 137, pp. 337–371). Springer.

Stevens, P., & Stirling, C. (1998). Practical model-checking using games. In B. Steffen (Ed.), *Proceedings of the 4th international conference on tools and algorithms for the construction and analysis of systems, TACAS'98, LNCS* (Vol. 1384, pp. 85–101). Springer.

Stirling, C. (1995). Local model checking games. In *Proceedings of the 6th conference on concurrency theory, CONCUR'95, LNCS* (Vol. 962, pp. 1–11). Springer.

Stirling, C., & Walker, D. (1991). Local model checking in the modal μ-calculus. *Theoretical Computer Science, 89*(1), 161–177.

Tarski, A. (1955). A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics, 5*, 285–309

ten Cate, B. (2005). *Model theory for extended modal languages*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands.

Vergauwen, B., & Lewi, J. (1993). A linear local model checking algorithm for CTL. In *Lecture Notes in Computer Science* (Vol. 715, pp. 447–461).

Vöge, J., & Jurdziński, M. (2000). A discrete strategy improvement algorithm for solving parity games. In *Proceedings of the 12th international conference on computer aided verification, CAV'00, LNCS* (Vol. 1855, pp. 202–215). Springer.