Hybrid Branching-Time Logics*

Volker Weber

Fachbereich Informatik, Universität Dortmund
44221 Dortmund, Germany
`volker.weber@udo.edu`

Abstract. Hybrid branching-time logics are introduced as extensions of $\mathcal{CTL}$-like logics with state variables and the downarrow-binder. Following recent work in the linear framework, only logics with a single variable are considered. The expressive power and the complexity of satisfiability of the resulting logics is investigated.

As main result, the satisfiability problem for the hybrid versions of several branching-time logics is proved to be **2EXPTIME**-complete. These branching-time logics range from strict fragments of $\mathcal{CTL}$ to extensions of $\mathcal{CTL}$ that can talk about the past and express fairness-properties. The complexity gap relative to $\mathcal{CTL}$ is explained by a corresponding succinctness result.

To prove the upper bound, the automata-theoretic approach to branching-time logics is extended to hybrid logics, showing that non-emptiness of alternating one-pebble Büchi tree automata is **2EXPTIME**-complete.

1 Introduction

Hybrid logics are extensions of modal logic that allow to refer to individual states of a model. They aim at extending the expressive power of modal logics, without losing their nice properties such as decidability. Hybrid logics have been researched quite intensively during the last years. Their applications range from verification to reasoning about semistructured data [10]. See [4] for a recent survey and an introduction to hybrid logic.

On the technical side, the aims of hybrid logic can be achieved by adding nominals and state variables, corresponding to the first-order concepts of constants and variables. Nominals are an additional kind of atomic symbols which are true in exactly one state in a model, and therefore name this state. Nominals are fixed with the model, whereas the assignment of states to state variables can be changed by quantification. To preserve the local perspective of modal logic, the quantifier considered the most is the downarrow-operator, first introduced in [12] and denoted ↓, which binds a state variable to the current state.

Satisfiability of hybrid ↓-languages is undecidable with respect to arbitrary Kripke-structures [2] and only non-elementarily decidable if the class of models is restricted to trees [16] or linear structures [11], i.e., to those models important in verification. These results initiated research on decidable fragments and fragments of lower complexity [20,18].

---

* An extended abstract of this paper was presented at the International Workshop on Hybrid Logics (HyLo 2007).

In [18], Schwentick and W. considered bounded-variable fragments of hybrid ↓-languages in the linear framework. While complexity of the two-variable fragment is already as bad as for the unbounded language, satisfiability of the one-variable fragment is **EXPSPACE**-complete. Furthermore, the one-variable fragment has the full expressive power of first-order logic.

The aim of this paper is to extend this successful approach to the branching-time framework. While the logic of [18] could also be interpreted over trees, we believe that hybrid extensions of classical branching-time logics like $\mathcal{CTL}$ are a more convenient formalism to reason about trees.

Our main result is that satisfiability for the one-variable-fragment of hybrid $\mathcal{ECTL}$+Past is **2EXPTIME**-complete. The lower bound is already achieved for the logic containing only the next **X** and future **F** modalities and is explained by a corresponding succinctness result. The upper bound is by a reduction to non-emptiness of alternating one-pebble Büchi tree automata, a problem that we prove to be **2EXPTIME**-complete as well. Furthermore, we study the expressive power of hybrid branching-time logics and show, for example, that the one-variable-fragment of hybrid $\mathcal{CTL}$ is strictly more expressive than $\mathcal{CTL}$.

Section 2 gives the basic notions of branching-time logics and introduces their hybrid extension. It also contains the definition of alternating one-pebble Büchi tree automata. Section 3 is concerned with the expressive power of hybrid branching-time logics, which are compared with classical branching-time logics and logics with the **N**-operator ("from now on") of [15]. The complexity and succinctness results can be found in Section 4, those on tree automata in Section 5. We give some directions for further research in Section 6.

2 Preliminaries

The basic definitions of branching-time logics and Büchi tree automata are presented in this section. As both formalism are defined with respect to infinite trees, we start by defining these structures.

Let $D=\{1,\ldots,k\}$ be a finite set of directions for some $k \in \mathbb{N}$. An *infinite D-tree* is a prefix-closed set $T \subseteq D^*$, i.e., whenever $x \cdot c \in T$ where $x \in D^*$ and $c \in D$, then also $x \in T$. The empty string ε is the *root* of T and for all $c \in D$, $x \cdot c$ is called a *child* of the node x. A *path* π in T is a prefix-closed minimal set $\pi \subseteq T$, such that for every $x \in \pi$, there is a unique $c \in D$ with $x \cdot c \in \pi$. We use "$\leq$" to denote the descendant-relation on T, i.e., $x < y$ if and only if y is a strict descendant of x. Note that this order is partial as nodes in different branches are incomparable. The branching degree $d(x)$ is the number of children of a node x. We only consider *k-ary trees*, where $d(x) = k$ for every node x, and refer to them as trees in the following.

A *labeled tree* over a finite alphabet Σ is a pair (T, V) where T is a tree and $V : T \to \Sigma$ assigns a symbol from Σ to every node of T. We are mainly interested in the case in which $\Sigma = 2^{\mathsf{PROP}}$ for some set PROP of propositions. Such trees, usually referred to as *computation trees*, result for example from the unwinding of Kripke structures (see, e.g., [13]). In the following, we identify (T, V) with T.

2.1 Branching-Time Logics

We briefly recall the basic notions of branching-time logic, starting from the logic $\mathcal{ECTL}$+Past which extends $\mathcal{CTL}$ in two ways: First, by adding the $\overset{\infty}{\mathbf{F}}$-modality for "infinitely often in the future" to express fairness properties. Secondly, by introducing past modalities $\mathbf{Y}$ for "previous" and $\mathbf{S}$ for "since".

$\mathcal{ECTL}$+Past-formulas are defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{EX}\varphi \mid \mathbf{E}\varphi\mathbf{U}\varphi \mid \mathbf{A}\varphi\mathbf{U}\varphi \mid \mathbf{E}\overset{\infty}{\mathbf{F}} \mid \mathbf{Y}\varphi \mid \varphi\mathbf{S}\varphi$$

where $p \in \mathsf{PROP}$. We use the usual abbreviations $\top$, $\bot$, $\varphi \vee \varphi$, $\varphi \to \varphi$, and

$$\mathbf{EF}\varphi := \mathbf{E}\top\mathbf{U}\varphi \qquad \mathbf{EG}\varphi := \neg\mathbf{AF}\neg\varphi \qquad \mathbf{A}\overset{\infty}{\mathbf{G}}\varphi := \neg\mathbf{E}\overset{\infty}{\mathbf{F}}\neg\varphi$$

$$\mathbf{AF}\varphi := \mathbf{A}\top\mathbf{U}\varphi \qquad \mathbf{AG}\varphi := \neg\mathbf{EF}\neg\varphi \qquad \mathbf{A}\overset{\infty}{\mathbf{F}}\varphi := \mathbf{AGAF}\varphi$$

$$\mathbf{AX}\varphi := \neg\mathbf{EX}\neg\varphi \qquad \mathbf{P}\varphi := \top\mathbf{S}\varphi \qquad \mathbf{E}\overset{\infty}{\mathbf{G}}\varphi := \mathbf{EFEG}\varphi$$

The semantics of $\mathcal{ECTL}$+Past-formulas is defined with respect to a computation tree T and a node n of T:

- $T, n \models p$ iff $p \in V(n)$
- $T, n \models \neg\varphi$ iff $T, n \not\models \varphi$
- $T, n \models \varphi \wedge \psi$ iff $T, n \models \varphi$ and $T, n \models \psi$
- $T, n \models \mathbf{EX}\varphi$ iff there exists a $c \in D$, such that $T, n \cdot c \models \varphi$
- $T, n \models \mathbf{E}\varphi\mathbf{U}\psi$ iff there exists a path π with $n \in \pi$, such that there is a descendant $n' \in \pi$ of n with $T, n' \models \psi$ and for all nodes $x \in \pi$ with $n \leq x < n'$, we have $T, x \models \varphi$
- $T, n \models \mathbf{E}\overset{\infty}{\mathbf{F}}\varphi$ iff there exists a path π with $n \in \pi$, such that there are infinitely many descendants $n' \in \pi$ of n with $T, n' \models \varphi$
- $T, n \models \mathbf{A}\varphi\mathbf{U}\psi$ iff for all paths π with $n \in \pi$, there is a descendant $n' \in \pi$ of n with $T, n' \models \psi$ and for all nodes $x \in \pi$ with $n \leq x < n'$, we have $T, x \models \varphi$
- $T, n \models \mathbf{Y}\varphi$ iff $T, n' \models \varphi$ with $n = n' \cdot c$ for some $c \in D$
- $T, n \models \varphi\mathbf{S}\psi$ iff there exists an ancestor n' of n, such that $T, n' \models \psi$ and for all nodes x with $n' < x \leq n$, we have $T, x \models \varphi$

Two formulas φ and ψ are *equivalent*, if $T, \varepsilon \models \varphi \iff T, \varepsilon \models \psi$ for all computation trees T, i.e., we only consider initial equivalence as we want to compare the expressive power of logics with and without past modalities [15].

We consider several fragments of $\mathcal{ECTL}$+Past and denote them by $\mathcal{B}(C)$, where C is the set of temporal operators allowed. To give some examples:

- $\mathcal{B}(\mathbf{X}, \mathbf{F})$ is the logic $\mathcal{UB}$ of [5],
- $\mathcal{B}(\mathbf{X}, \mathbf{U})$ is the well known logic $\mathcal{CTL}$ [7],
- $\mathcal{B}(\mathbf{X}, \mathbf{U}, \overset{\infty}{\mathbf{F}})$ was introduced in [8] as $\mathcal{ECTL}$, and
- $\mathcal{B}(\mathbf{X}, \mathbf{U}, \mathbf{Y}, \mathbf{S})$ is $\mathcal{PCTL}$, i.e., $\mathcal{CTL}$ +Past.

In all these logics, future temporal operators occur only immediately in the scope of the path quantifiers $\mathbf{E}$ and $\mathbf{A}$. Opposed to this, the branching time logic $\mathcal{CTL}^*$ from [8] allows Boolean combinations and nesting of these operators.

2.2 Hybrid Branching-Time Logics

We extend branching-time logics with hybrid machinery along the lines of [18]. I.e., we use only one state variable x and consider only a single nominal *root*.

Definition 2.1. Given a set $C \subseteq \{\mathbf{X}, \mathbf{F}, \mathbf{U}, \overset{\infty}{\mathbf{F}}, \mathbf{Y}, \mathbf{P}, \mathbf{S}\}$ of modalities and a branching-time logic $\mathcal{B}(C)$, the formulas of the corresponding *hybrid branching-time logic* $\mathcal{HB}(C)$ are those of $\mathcal{B}(C)$ and additional

$$\downarrow x.\varphi \mid x \mid @_x\varphi \mid root \mid @_{root}\varphi,$$

where φ is a $\mathcal{HB}(C)$-formula and x is the only state variable.

The semantics of hybrid branching-time formulas is defined with respect to a computation tree T and two nodes n, m of T, where n is the current node and m is the node assigned to the state variable x:

$$
\begin{aligned}
T, n, m &\models\, \downarrow x.\varphi &&\text{iff } T, n, n \models \varphi & \qquad T, n, m &\models x &&\text{iff } n = m \\
T, n, m &\models @_x\varphi &&\text{iff } T, m, m \models \varphi & \qquad T, n, m &\models root &&\text{iff } n = \varepsilon \\
T, n, m &\models @_{root}\varphi &&\text{iff } T, \varepsilon, m \models \varphi
\end{aligned}
$$

and the semantics of classical branching-time logic are extended in the obvious way, i.e., the state variable is not affected.

A formula φ is called *satisfiable* if there is a computation tree T and nodes n, m such that $T, n, m \models \varphi$.

Remark 2.2 (Using hybrid machinery). We give two examples on how hybrid branching-time logics work. The reader will find both patterns again in the proofs given in this paper.

Hybrid branching-time logics can reason about the past without using past modalities. The past formula $\mathbf{P}\varphi$ can be expressed as $\downarrow x.@_{root}\mathbf{EF}(\mathbf{EF}x \wedge \varphi)$, for example. This illustrates how a finite prefix of a path can be fixed.

Moreover, they can easily compare two nodes. The property that there are two different nodes in a tree that agree on the propositions $p_1, \ldots, p_n$, can be expressed as $\mathbf{EF}(\downarrow x.@_{root}\mathbf{EF}(\neg x \wedge \bigwedge_{i=1}^{n} p_i \leftrightarrow @_x p_i))$.

Remark 2.3 (Bisimulations). Bisimulation equivalence is not respected by hybrid branching-time logics: We can distinguish two isomorphic subtrees by naming the root of one of those subtrees. But they respect *hybrid one-bisimulations* defined in [3] and successfully applied in [18] in the linear framework.

2.3 Tree Automata

The following basic notions about Büchi automata on infinite trees are based on the definitions in [22] and [23]. For a more general introduction to automata on infinite trees, we refer to [21].

A *non-deterministic Büchi tree automaton* A is a tuple $(Q, \Sigma, q^0, \delta, F)$, where Q is a finite set of states, Σ is a finite alphabet, $q^0 \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, and $\delta : Q \times \Sigma \to 2^{Q^k}$ is a transition function. Whenever

A is in state q at a node x, it non-deterministically chooses a k-tuple $(q_1, \ldots, q_k)$ of states from $\delta(q, V(x))$ and moves to node $x \cdot i$ in state q_i for each $i = 1, \ldots, k$.

A *run* r of A on a Σ-labeled tree (T, V) is a Q-labeled tree (T, V'), such that the root is labeled by the initial state and the transition rules are respected, i.e., if a node x is labeled q and its children are labeled $q_1, \ldots, q_k$, then $(q_1, \ldots, q_k) \in \delta(q, V(x))$. A run r is *accepting* if $lim(\pi) \cap F \neq \emptyset$ for every infinite path π of r, where $lim(\pi)$ is the set of states occurring infinitely on π. A labeled tree (T, V) is *accepted* by A if there is an accepting run of A on (T, V). The *language* of A is the set of trees accepted by A and denoted $L(A)$.

Proposition 2.4 ([17]). *Non-emptiness of non-deterministic Büchi tree automata can be decided in quadratic time.*

Alternating one-pebble Büchi tree automata generalize this concept in three ways. First, they are two-way automata, i.e., they can also move upward in the tree. Additionally, they can drop a pebble at a position in the tree and lift the pebble again if they are at the position where the pebble was placed. In other words, these automata can mark a position to find it again after moving away. Finally, they can universally and existentially branch into several independent sub-computations.

More formally, an *alternating one-pebble Büchi tree automaton* is a tuple $A = (Q, \Sigma, q^0, \delta, F)$, such that Q is a finite set of states, Σ is a finite alphabet, $q^0 \in Q$ is the initial state, $F \subseteq Q$ is the set of accepting states, and

$$\delta : Q \times \Sigma \to (Q \times \{\text{drop,lift}\}) \cup \mathcal{B}^+([k] \times Q)$$

is a transition function.

In this definition, we use $[k] := \{-1, 0, 1, \ldots, k\}$ to give the direction of a move of the automaton and $\mathcal{B}^+(X)$ to denote the set of positive Boolean formulas over X, i.e., formulas built from X by $\wedge$ and $\vee$ including $\top$ and $\bot$. Note that such an automaton can send several subcomputations into the same direction, but does not need to go into every direction.

A *configuration* $(q, x, y) \in Q \times D^* \times (D^* \cup \{\bot\})$ of A consists of a state, the current position in the tree, and the position of the pebble, where "$\bot$" means that the pebble is not placed.

A *run* r of A on an infinite labeled k-ary tree (T, V) is a possibly infinite tree (T', V') whose nodes are labeled by configurations of A. This tree must be compatible with the transition function. For example, for every node $v \in T'$ labeled by a state $(q, x \cdot c, y)$,

- if $\delta(q, V(x \cdot c)) = (q', \text{drop})$ and $y = \bot$, then v has a child labeled with $(q, x \cdot c, x \cdot c)$, otherwise, i.e. if $y \neq \bot$, the transition cannot be applied;
- if $\delta(q, V(x \cdot c)) = (q', \text{lift})$, then v has a child $(q', x \cdot c, \bot)$ if $x \cdot c = y$, otherwise the transition cannot be applied;
- if $\delta(q, V(x \cdot c)) = (1, q') \wedge (-1, q'')$, v has children labeled by $(q', x \cdot c \cdot 1, y)$ and (q'', x, y);
- if $\delta(q, V(x \cdot c)) = (0, q') \vee (2, q'')$, then v has a child labeled by $(q', x \cdot c, y)$ or a child labeled $(q'', x \cdot c \cdot 2, y)$.

A run is *accepting* if every infinite path contains infinitely many configurations with states from F. Acceptance of A is defined as usual.

3 Expressivity

We examine the expressive power of hybrid branching-time logics. By Remark 2.3, these logics are strictly more expressive than their classical counterparts. In the first part of this section, we give two examples where hybrid machinery is used to cover even more expressive classical branching-time logics. These results are in contrast to [18], where it was shown that the hybrid version of $\mathcal{LTL}$ is expressively equivalent to $\mathcal{LTL}$. The second part compares branching-time logics with the **N**-operator to hybrid branching-time logics.

3.1 Capturing Classical Branching-Time Logics

Adding hybrid machinery to $\mathcal{B}(\mathbf{X}, \mathbf{F})$ results in a strictly more expressive logic.

Theorem 3.1. $\mathcal{HB}(\mathbf{X}, \mathbf{F})$ *is strictly more expressive than* $\mathcal{B}(\mathbf{X}, \mathbf{F})$.

Proof. It is known from [15] that the extension of $\mathcal{B}(\mathbf{X}, \mathbf{F})$ with one of the past modalities **S** or **Y** is strictly more expressive than $\mathcal{B}(\mathbf{X}, \mathbf{F})$. We show that both **S** and **Y** can be expressed in $\mathcal{HB}(\mathbf{X}, \mathbf{F})$.

For the **Y** modality, the idea is to fix the current node by naming it x and then to jump to the root and move forward to the node where $\mathbf{EX}x$ holds. This node is the unique predecessor of the node named x. If the latter node is already the root, no predecessor exists and the following formula evaluates to false.

$$\mathbf{Y}\varphi \equiv \downarrow x.@_{root}\mathbf{EF}((\mathbf{EX}x) \wedge \varphi)$$

The Since-modality can be replaced in a similar way:

$$\varphi\mathbf{S}\psi \equiv \downarrow x.@_{root}\mathbf{EF}(\mathbf{EF}x \wedge \psi \wedge (x \vee \mathbf{EX}(\mathbf{EF}x \wedge \mathbf{AG}(\mathbf{EF}x \rightarrow \varphi)))),$$

respecting that ψ either holds at the current or at some previous node. $\square$

As both formulas given in the previous proof are of linear size in the length of the past-formulas, we obtain the following intensification of Theorem 3.1.

Corollary 3.2. *There is a linear translation from* $\mathcal{B}(\mathbf{X}, \mathbf{F}, \mathbf{Y}, \mathbf{S})$ *to* $\mathcal{HB}(\mathbf{X}, \mathbf{F})$.

This shows that every hybrid branching-time logic containing **X** and **F** can refer to the past. In particular, the hybrid version of $\mathcal{CTL}$ captures the extension of $\mathcal{CTL}$ with past modalities, which is known to be strictly more expressive than the pure future logic [15]. As hybrid branching-time logics do not respect bisimulation-equivalence, this inclusion is strict.

Theorem 3.3. $\mathcal{HB}(\mathbf{X}, \mathbf{U})$ *is strictly more expressive than* $\mathcal{PCTL}$.

The bisimulation argument even shows that there cannot be a translation from hybrid branching-time logics into $\mathcal{PCTL}^*$, since the latter can express only bisimulation-invariant properties.

On the other hand, we conjecture that it is not possible to express the $\mathcal{ECTL}$-formula $\mathbf{E}\overset{\infty}{\mathbf{F}}p$ in $\mathcal{HB}(\mathbf{X}, \mathbf{U})$. In this case, $\mathcal{HB}(\mathbf{X}, \mathbf{U})$ and $\mathcal{CTL}^*$ are incomparable with respect to expressive power.

Finally, hybrid branching-time logics are obviously fragments of Monadic Path Logic ($\mathcal{MPL}$), the fragment of $\mathcal{MSO}$ where set-quantification is restricted to paths. This inclusion can be proved to be strict by observing that hybrid branching-time logics respect hybrid one-bisimulations as defined in [3].

3.2   Expressing "From Now On"

The temporal operator $\mathbf{N}$ for "from now on" was introduced by Laroussinie and Schnoebelen to branching-time logics with past [15]. The semantics of $\mathbf{N}$ is given by: $T, n \models \mathbf{N}\varphi$ iff $T', \varepsilon \models \varphi$, where $T' = \{m \in \mathbb{N}^* \mid n \cdot m \in T\}$ is the subtree of T rooted at n. That is, $\mathbf{N}$ allows to *forget about the past*.

In [15] the authors provide several results on whether $\mathbf{N}$ adds expressive power to branching-time logics with past. E.g., it does for $\mathcal{B}(\mathbf{X}, \mathbf{F}, \mathbf{Y}, \mathbf{P})$ but does not for $\mathcal{PCTL}$ and $\mathcal{PCTL}^*$. Moreover, they argue that $\mathbf{N}$ offers a more convenient way to describe some properties in branching-time logics with past (see [15] for an example), which is partially attributed to the succinctness of logics with the $\mathbf{N}$-operator.[1]

The following proposition shows that hybrid branching-time logics offer at least the same convenience and are at least as succinct as the logics including $\mathbf{N}$.

Proposition 3.4. *For every set of modalities $C \subseteq \{\mathbf{X}, \mathbf{F}, \mathbf{U}, \overset{\infty}{\mathbf{F}}, \mathbf{Y}, \mathbf{P}, \mathbf{S}\}$, there is a linear translation from $\mathcal{B}(\mathbf{N}, C)$ to $\mathcal{HB}(\mathbf{P}, C)$.*

Proof. Given a $\mathcal{B}(\mathbf{N}, C)$-formula φ, we obtain an equivalent $\mathcal{HB}(\mathbf{P}, C)$-formula $\downarrow x.\varphi'$ by substituting $\mathbf{N}$ by the downarrow-operator and guarding all past modalities. I.e., φ' results from φ by applying the following rules once for every past modality and every $\mathbf{N}$-operator:

$$\mathbf{N}\psi \to \downarrow x.\psi \qquad\qquad \mathbf{P}\psi \to \mathbf{P}(\mathbf{P}x \wedge \psi)$$
$$\mathbf{Y}\psi \to \mathbf{Y}(\mathbf{P}x \wedge \psi) \qquad\qquad \varphi\mathbf{S}\psi \to \varphi\mathbf{S}(\mathbf{P}x \wedge \psi)$$

Requiring $\mathbf{P}$ is only a restriction if $\mathbf{Y}$ is the only past modality in C. $\square$

As we show in the next section, we cannot add $\mathbf{N}$ on top of hybrid branching-time logics without blowing up the complexity of satisfiability non-elementarily. Intuitively, this is because $\mathbf{N}$ can play the role of a second state variable, therefore enabling us to talk about three points at the same time: the new root created by $\mathbf{N}$, the state named x, and the current state.

But the reader should be warned not to think of $\mathbf{N}$ as a kind of state variable, since the ability to name a state and then to talk about its past is crucial to most results in this paper.

---

[1] To the best of our knowledge, this succinctness gap has so far only been proved for the case of linear temporal logic in [14]. Succinctness and complexity for branching-time logics with $\mathbf{N}$ seem to be open problems.

4 Complexity of the Satisfiability Problem

The main motivation behind the one-variable approache to hybrid branching-time logics is to tame the complexity of the satisfiability problem. This section shows that this approache was successful by providing a **2EXPTIME**-completeness result for satisfiability of several hybrid branching-time logics.

The proof of the lower complexity bound for $\mathcal{HB}(\mathbf{X}, \mathbf{F})$ is by a reduction from the 2^n-corridor tiling game. We first define the 2^n-*corridor tiling problem*. An instance $I = (T, H, V, n)$ of this problem consists of a finite set T of tile types, horizontal and vertical constraints $H, V \subseteq T \times T$, and a number n given in unary. The task is to decide, whether T tiles the $2^n \times m$-corridor for some m, respecting the constraints H and V and some border constraints, especially on the top row to be reached.

The 2^n-*corridor tiling game* is played by two players E and A on an instance I of the 2^n-corridor tiling problem. The players alternately place tiles starting with player E and following the constraints H and V, as the opponent wins otherwise. E wins the game if the required top row is reached. To decide whether E has a winning strategy in such a game is complete for **2EXPTIME** [6].

Proposition 4.1. *Satisfiability of $\mathcal{HB}(\mathbf{X}, \mathbf{F})$ is hard for* **2EXPTIME**.

Proof. Let $I = (T, H, V, n)$ be an instance of the 2^n-corridor tiling problem. We build an $\mathcal{HB}(\mathbf{X}, \mathbf{F})$-formula φ_I of size polynomial in $|I|$ that is satisfiable if and only if player E has winning strategy in the tiling game on I.

Such a winning strategy is a finite T-labeled tree whose levels alternately correspond to moves of E and A. A node corresponding to a move of E, as the root for example, has one child for every possible next move of A. Nodes representing moves of A have only one child: *the best move E can make*. In order for the strategy to be winning, every path in this tree has to correspond to a correct tiling reaching the required top row.

The formula φ_I consists of two parts. The first part describes an encoding of a winning strategy, using a numbering of the states belonging to one row of the tiling as shown in Figure 1. Numbers are encoded by n propositions, one for each digit. The second part basically contains the conditions posed by H and V.

To make this more precise, φ_I is the conjunction of the two formulas φ_{struc} and φ_{tiles}. The first formula, φ_{struc}, starts by separating the lines of the tiling by an additional state labeled by the proposition symbol $q_\#$ and marking the
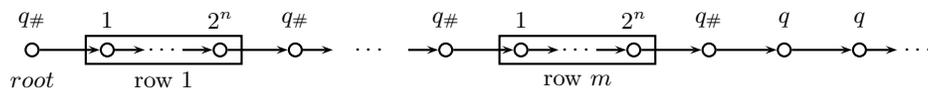

Fig. 1. A path in the encoding of a winning strategy for the 2^n-corridor tiling game with m rows.

states beyond the encoding by the proposition symbol q.

$$\varphi_{struc} = root \wedge q_\# \wedge \neg q \wedge (\bigwedge_{i=0}^{n-1} \neg q_i) \wedge \mathbf{AG}((\neg q \wedge \neg q_\#) \vee ((q \leftrightarrow \neg q_\#) \wedge \bigwedge_{i=0}^{n-1} \neg q_i))$$

$$\wedge \mathbf{AF}(q_\# \wedge \mathbf{AG}q) \wedge \mathbf{AG}(q \rightarrow \mathbf{AG}q) \wedge \varphi_{num}$$

To get a correct numbering of the states representing one row of the tiling, φ_{num} requires every state to have only properly numbered direct successors. This numbering is required to check the vertical constraints.

$$\varphi_{num} = \mathbf{AG}([q_\# \rightarrow (\mathbf{AX}(\neg q \wedge \neg q_\# \wedge \bigwedge_{i=0}^{n-1} \neg q_i) \vee \mathbf{AX}q)]$$

$$\wedge[(\neg q \wedge \neg q_\#) \rightarrow ((\bigwedge_{i=0}^{n-1} q_i \wedge \mathbf{AX}q_\#) \vee \nu)])$$

$$\nu = \downarrow x.\mathbf{AX} \bigvee_{i=0}^{n-1} (\bigwedge_{j<i}(q_j \leftrightarrow @_x \neg q_j) \wedge q_i \wedge @_x \neg q_i \wedge \bigwedge_{j>i}(q_j \leftrightarrow @_x q_j))$$

The second part of φ_I expresses that every state corresponding to a move of one of the players is labeled by exactly one tile, using proposition symbols p_t to represent the tiles, that the conditions in H and V are respected, and that all possible moves of A are represented.

$$\varphi_{tiles} = \mathbf{AG}([\neg q \wedge \neg q_\#] \rightarrow [\bigvee_{t \in T}(p_t \wedge \bigwedge_{t \neq t' \in T} \neg p_{t'}) \wedge \theta_H \wedge \theta_V \wedge \theta_A])$$

$$\theta_H = \neg \bigwedge_{i=0}^{n-1} q_i \rightarrow \bigwedge_{t \in T}(p_t \rightarrow \mathbf{AX} \bigvee_{(t,t') \in H} p_{t'})$$

$$\theta_V = \downarrow x.@_{root}\mathbf{AG}([\mathbf{EXEF}x \wedge \neg \mathbf{EF}(q_\# \wedge \mathbf{EXEF}(q_\# \wedge \mathbf{EF}x))$$

$$\wedge \bigwedge_{i=0}^{n-1}(q_i \leftrightarrow @_x q_i)] \rightarrow \bigwedge_{t \in T}(p_t \rightarrow \bigvee_{(t,t') \in V} @_x p_{t'})))$$

$$\theta_A = \neg q_0 \rightarrow \bigwedge_{t \in T}(p_t \rightarrow \bigwedge_{(t,t') \in H}[\mathbf{EX}p_{t'} \vee \downarrow x.@_{root}\mathbf{AG}([\mathbf{EXEF}x \wedge \neg \mathbf{EF}(q_\#$$

$$\wedge \mathbf{EXEF}(q_\# \wedge \mathbf{EF}x) \wedge \bigwedge_{i=0}^{n-1}(q_i \leftrightarrow @_x q_i)] \rightarrow \mathbf{EX}[\mathbf{EF}x \wedge \bigvee_{(t'',t') \notin V} p_{t''}])]])$$

We omit formulas for the border constraints, which are straightforward. □

Before we proceed with the upper bound, we show that the lower bound is due to the succinctness of hybrid formulas.

Theorem 4.2. *The succinctness of* $\mathcal{HB}(\mathbf{F})$ *with respect to* $\mathcal{CTL}$ *is* $O(n)!$, *i.e., there is a* $\mathcal{HB}(\mathbf{F})$*-formula of length* $O(n)$ *such that every equivalent* $\mathcal{CTL}$*-formula is at least of length* $O(n)!$.

Proof. We consider the $\mathcal{CTL}^+$-formula $\mathbf{E}(\mathbf{F}p_1 \wedge \mathbf{F}p_2 \wedge \cdots \wedge \mathbf{F}p_n)$, expressing that there exists a path such that each of the propositions $p_1, \ldots, p_n$ holds at some node on the path. Adler and Immerman proved that one requires a formula of size $O(n)!$ to express this property in $\mathcal{CTL}$ [1]. The following $\mathcal{HB}(\mathbf{F})$-formula has only size $O(n)$.

$$\mathbf{EF}(\downarrow x.@_{root} \bigwedge_{i=1}^{n} \mathbf{EF}(p_i \wedge \mathbf{EF}x))$$

The crucial point is that this property depends only on a finite prefix of the path, which can be fixed by naming its last state x. □

The proof of the upper complexity bound for satisfiability of hybrid branching-time logics uses the automata-theoretic approach to branching-time logics (see [24] and references therein), extended to the hybrid framework.

Before we go on, we observe that nesting of the $\downarrow$-operator can be avoided.

Lemma 4.3. *For every $\mathcal{HB}(\mathbf{X}, \mathbf{U}, \overset{\infty}{\mathbf{F}}, \mathbf{Y}, \mathbf{S})$-formula φ, there is an equivalent formula ψ of length $O(|\varphi|)$ without nested occurrences of the $\downarrow$-operator.*

Proof. We add, for each sub-formula $\theta = \downarrow x.\xi$ of φ, a new proposition p_θ. In a bottom-up fashion, we replace every occurrence of a formula θ by p_θ and add to φ one conjunct $\mathbf{AG}(p_\theta \leftrightarrow \theta')$, for every θ. Here, θ' results from θ by replacing all strict sub-formulas $\downarrow x.\chi$ by the respective proposition. □

We can now prove the main theorem of this paper.

Theorem 4.4. *For every set of temporal operators $C \subseteq \{\mathbf{U}, \mathbf{Y}, \overset{\infty}{\mathbf{F}}, \mathbf{P}, \mathbf{S}\}$, the satisfiability problem for $\mathcal{HB}(\mathbf{X}, \mathbf{F}, C)$ is complete for $\mathbf{2EXPTIME}$.*

Proof. The lower bound was proved in Proposition 4.1.

The proof of the upper bound is a extension of a proof in [22], constructing an alternating Büchi tree automaton for a given $\mathcal{CTL}$-formula.

Given an $\mathcal{HB}(\mathbf{X}, \mathbf{U}, \overset{\infty}{\mathbf{F}}, \mathbf{Y}, \mathbf{S})$-formula φ without nested occurrences of the $\downarrow$-operator, we build an alternating one-pebble Büchi tree automaton $A_\varphi = (Q, \Sigma, q^0, \delta, F)$, with $\Sigma = 2^{\mathsf{PROP}}$, such that φ holds at the root of some Σ-labeled tree (T, V) if and only if A_φ accepts this tree. This reduces the satisfiability problem for $\mathcal{HB}(\mathbf{X}, \mathbf{U}, \overset{\infty}{\mathbf{F}}, \mathbf{Y}, \mathbf{S})$ to non-emptiness of alternating one-pebble Büchi tree automata. The latter problem is proved to be in $\mathbf{2EXPTIME}$ in Section 5.

In the following, we denote the *dual* of a formula ψ by $\overline{\psi}$. It is obtained from ψ by switching $\wedge$ and $\vee$, and by negating all other maximal subformulas (we identify $\neg\neg\psi$ with ψ), e.g., $\overline{x \vee (\neg x \wedge \mathbf{EF}p)} = \neg x \wedge (x \vee \neg\mathbf{EF}p)$ (cf. [22]).

The set Q of states is based on the Fisher-Ladner-closure of φ, consisting of the subformulas of φ and their duals. Additionally, the formula $(\mathbf{EXE}\overset{\infty}{\mathbf{F}}\psi) \wedge \psi$ and all its subformulas are included for every subformula $\mathbf{E}\overset{\infty}{\mathbf{F}}\psi$ of φ. The initial state q^0 is φ. The set F of accepting states contains $\top$ and all formulas of the form $\neg\mathbf{E}(\chi\mathbf{U}\psi)$, $\neg\mathbf{A}(\chi\mathbf{U}\psi)$, and $(\mathbf{EXE}\overset{\infty}{\mathbf{F}}\psi) \wedge \psi$ from Q. The transition function δ is defined by induction on the formula structure:

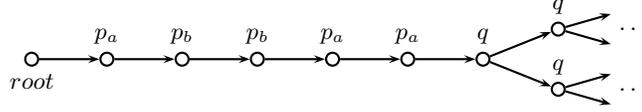**Fig. 2.** The tree used to represent the string *abbaa* in the proof of Theorem 4.5.

$$\delta(\top,\sigma) = (0,\top) \qquad\qquad \delta(p,\sigma) = (0,\top) \qquad \text{if } p \in \sigma$$

$$\delta(\neg\psi,\sigma) = \overline{\delta(\psi,\sigma)} \qquad\qquad \delta(\psi \wedge \xi,\sigma) = (0,\psi) \wedge (0,\xi)$$

$$\delta(x,\sigma) = (\top,\text{lift}) \qquad\qquad \delta(\mathbf{E}\overset{\infty}{\mathbf{F}}\varphi,\sigma) = \bigvee_{i=1}^{k}(i,\mathbf{E}\overset{\infty}{\mathbf{F}}\varphi) \vee (0,(\mathbf{EXE}\overset{\infty}{\mathbf{F}}\varphi) \wedge \varphi)$$

$$\delta(\downarrow x.\psi,\sigma) = (\psi,\text{drop}) \qquad \delta(\mathbf{E}(\chi\mathbf{U}\psi),\sigma) = (0,\psi) \vee ((0,\chi) \wedge \bigvee_{i=1}^{k}(i,\mathbf{E}(\chi\mathbf{U}\psi)))$$

$$\delta(\mathbf{EX}\psi,\sigma) = \bigvee_{i=1}^{k}(i,\psi) \qquad \delta(\mathbf{A}(\chi\mathbf{U}\psi),\sigma) = (0,\psi) \vee ((0,\chi) \wedge \bigwedge_{i=1}^{k}(i,\mathbf{A}(\chi\mathbf{U}\psi)))$$

$$\delta(\mathbf{Y}\psi,\sigma) = (-1,\psi) \qquad\qquad \delta(\chi\mathbf{S}\psi,\sigma) = (0,\psi) \vee ((0,\chi) \wedge (-1,\chi\mathbf{S}\psi))$$

where $\sigma \in \Sigma$, $p \in \mathsf{PROP}$, and the notion of a dual is extended to δ in the obvious way, e.g., $\overline{\delta(\mathbf{EX}\psi,\sigma)} = \bigwedge_{i=1}^{k}(i,\overline{\psi})$.

The result then follows from Theorem 5.1. $\qquad\qquad\qquad\qquad\qquad\qquad$ □

This result is optimal with respect to the number of state variables available. We have shown in Section 3.2 that the **N**-operator can be simulated by a state variable, and therefore be seen as a "weak" kind of variable. In the following, we show that adding the **N**-operator to hybrid branching-time logics causes a non-elementary blow-up in complexity.

We have to be a bit careful when adding the **N**-operator to hybrid branching-time logics. First, what is the semantics of a formula of the form $@_{root}\psi$ in the scope of an **N**-operator? As the **N** was introduced to forget about the past, the most natural thing is to define that this formula jumps to the new root created my the **N**-operator. While this is minor if past modalities are available, it is the only reasonable choice for pure future hybrid branching-time logics.

The second difficulty is that the state variable might be bound to some state in the past. In order not to unbind the variable, we assume that in this case the assignment is updated to the current state, i.e., to the new root. But this situation does not occur in the following proof.

Theorem 4.5. *The satisfiability problem for $\mathcal{HB}(\mathbf{X},\mathbf{F},\mathbf{N})$ has non-elementary complexity.*

Proof. We give a reduction from the non-emptiness problem for star-free expressions built from union, concatenation, and negation. This problem is known to have non-elementary complexity [19]. With a string of length i over an alphabet Σ we associate a tree whose first $i + 1$ nodes have only one child. All states beyond carry the label q as shown in Figure 2.

The following formula ψ holds at the root if and only if the tree is an encoding of a string, e.g., every state belonging to the string is labeled by exactly one p_σ.

$$\psi = \mathbf{EF}((q \wedge \mathbf{AG}q)\wedge \downarrow x.@_{root}\mathbf{AG}(\mathbf{EFEX}x \rightarrow (\neg q \wedge \bigvee_{\sigma\in\Sigma}(p_\sigma \wedge \bigwedge_{\sigma\neq\sigma'\in\Sigma}\neg p_{\sigma'}))))$$

$$\wedge\neg q \wedge \mathbf{AG}(\neg q \rightarrow \mathbf{EX}\downarrow x.@_{root}\mathbf{AG}(\mathbf{EX}x \rightarrow \mathbf{AX}x))$$

We map every star-free expression α to a formula

$$\varphi_\alpha = \psi \wedge \mathbf{EF}(\neg q \wedge \mathbf{AX}q \wedge \downarrow x.@_{root}\alpha'),$$

where α' is inductively defined as follows:

$$
\begin{array}{ll}
\varepsilon' = x & \sigma' = \mathbf{EX}(x \wedge p_\sigma) \quad \text{, for all } \sigma \in \Sigma \\
\emptyset' = \bot & (\alpha \cdot \beta)' = \mathbf{EF}(\mathbf{EF}x \wedge \downarrow x.@_{root}\alpha' \wedge \mathbf{N}\beta') \\
(\neg\alpha)' = \neg\alpha' & (\alpha \cup \beta)' = \alpha' \vee \beta'
\end{array}
$$

The idea is that x is always used to mark the end of the substring which is matched with respect to a star-free (sub-)expression while its beginning is at the child of the root "created" by $\mathbf{N}$. □

5 Non-emptiness of Alternating One-Pebble Tree Automata

In this section, we show that the non-emptiness problem for alternating Büchi tree automata with one pebble is **2EXPTIME**-complete. The proof is based on [18] where **EXPSPACE**-completeness for the string case is shown.

Theorem 5.1. *Non-emptiness of alternating one-pebble Büchi tree automata is complete for* **2EXPTIME**.

To simplify the presentation of the proof of the upper bound, we assume that we do not have arbitrary positive Boolean combinations on the right-hand side of a transition rule, but only either disjunctions or conjunctions. This is equivalent to the more general notion used before. A configuration is called existential if the matching transition rule contains a disjunction and universal if it contains a conjunction.

A run r of an alternating Büchi tree automaton A with one pebble is an infinite in which the nodes are labeled with configurations (q, x, y), where q is the state, x is a node of the tree and j is the position of the pebble ($\bot$ if the pebble is not placed). We are interested in runs of the following structure. A run is *homogeneous* if, for every existential configuration (q, x, y), all nodes of r labeled with configuration (q, x, y) have the same configuration at their child.

Note that the configuration graph of A on a tree T can be seen as the arena of a two-player game with a Büchi winning condition. Thus, from the existence of memoryless winning strategies in such games [9] (see also [25]), it follows that if A has an accepting run on T, then it also has a homogeneous accepting run.

We show that for each alternating one-pebble Büchi tree automaton there is an equivalent non-deterministic Büchi tree automaton of double exponential size. The basic idea is to simulate an accepting homogeneous run of the alternating automaton by running the non-deterministic Büchi automaton constructed in [18] for the string case along every branch of the tree.

When arguing about runs, we will make use of the following version of König's Lemma.

Theorem 5.2 (König's Lemma). *If in a tree each node has only finitely many children and there are nodes of arbitrary depth, then the tree has an infinite path.*

Proof (of Theorem 5.1). Hardness follows from Proposition 4.1 via the translation presented in the proof of Theorem 4.4.

For the upper bound, we show that, for each alternating one-pebble Büchi tree automaton A, there is an equivalent non-deterministic Büchi tree automaton B of size $|\Sigma_A| \cdot 2^{2^{O(|Q_A|)}}$, which can be constructed from A in space exponential in $|Q_A|$. The result then follows by Proposition 2.4. As already indicated above, B checks, on input T, whether A has a homogeneous accepting run r on T.

A run is *not* accepting if and only if it has a non-accepting path. Thus, B checks that r has no non-accepting path. The non-accepting paths can be classified as follows. First, a path π can be *bounded*, i.e., there is some $m \in \mathbb{N}$ such that all nodes of T occurring in the labels along π are at most of depth m, otherwise, we call π *unbounded*.

There are two kinds of unbounded non-accepting paths:

(1a) At some point, the automaton A drops the pebble at some node y and never lift it again. In this case, all further configurations are of the form (q, x, y), for some q, x.

(1b) Otherwise, the path has infinitely many configurations of the form $(q, x, \bot)$.

In both cases x can be arbitrarily deep in T.

Likewise, there are two kinds of bounded non-accepting paths.

(2a) The first kind drops the pebble at some node y and never lifts it again. Thus, there are q and x such that (q, x, y) occurs infinitely often on π and there is a subpath from configuration (q, x, y) to (q, x, y) which does not visit any accepting state, does not lift the pebble, and does not visit any strict descendant of x.

(2b) The other kind of bounded paths has infinitely many configurations of the form $(q, x, \bot)$, hence there is again a maximum x and a state q such that $(q, x, \bot)$ occurs infinitely often and only finitely many nodes have a configurations $(p, x', \bot)$ with $x' > x$. Therefore, there is a subpath from configuration $(q, x, \bot)$ to $(q, x, \bot)$ which does not visit any accepting state and does not visit any descendant of x without having the pebble placed before.

Note that the node x is not unique in both cases. Therefore, B will check the existence of such path at every possible node x of T.

In the following we describe the information that B maintains in order to check that r has only accepting paths.

For each x, let S_x be the set of states q for which $(q, x, \bot)$ occurs in r. We consider two kinds of *upward paths* (*downward paths* are defined accordingly):

− paths starting from a configuration $(p, x, \bot)$ and ending in a configuration $(q, x, \bot)$ without an intermediate configuration $(p', x', \bot)$ with $x < x'$ (intermediate configurations (p', x', y) with $x \not< y$ and $x < x'$ are allowed);

− paths starting from a configuration (p, x, y) and ending in a configuration (q, x, y), without any intermediate lifting of the pebble and without any intermediate configurations (p', x', y) with $x < x'$.

For each node x, we denote by U_x the subset of $Q \times Q \times \{+, \exists\}$, such that

- $(p, q, +) \in U_x$ if and only if all upward paths of r from $(p, x, \bot)$ to $(q, x, \bot)$ visit an accepting state, and
- $(p, q, \exists) \in U_x$ if and only if r has a upward path from $(p, x, \bot)$ to $(q, x, \bot)$,

and by D_x the subset of $Q \times Q \times \{+, \exists\}$, such that

- $(p, q, +) \in D_x$ if and only if all downward paths[2] of r from (p, x, ε) to (q, x, ε) visit an accepting state, and
- $(p, q, \exists) \in D_x$ if and only if r has a downward path from (p, x, ε) to (q, x, ε).

It should be noted that in the definition of D_x, the actual position of the pebble does not matter, as long as it is not a descendant of x. The reader should also observe the asymmetry between the U_x and the D_x. The U_x only concern sub-computations from a configuration without pebble, the D_x only from a configuration with pebble.

Furthermore, B uses the following sets which are parametrized by the current position y of the pebble. Let, for each node x and each $y \not> x$, $S_{x,y}$ be the set of states p such that (p, x, y) occurs in r. Likewise, let $U_{x,y}$ be the subset of $Q \times Q \times \{+, \exists\}$ such that

- $(p, q, +) \in U_{x,y}$ if and only if all upward paths of r from (p, x, y) to (q, x, y) visit an accepting state and
- $(p, q, \exists) \in U_{x,y}$ if and only if r has a upward path from (p, x, y) to (q, x, y).

Recall that upward paths from a configuration (p, x, y) never lift the pebble.

Additionally, B uses sets R_x, $R_{x,y}$ and D'_i which will be defined below. For each x, we let X_x be the set $\{(S_{x,y}, U_{x,y}, R_{x,y}) \mid y \not> x\}$. Finally, for each x, let the characteristic vector C_x of position x be $(S_x, U_x, D_x, D'_x, R_x, X_x)$. The intended state of B at node x is basically $(C_{x \cdot -1}, C_x)$.

The sets of the form $S_x, S_{x,y}, D_x$ and the transitions of A are guessed by B and the remaining information can be determined from it. It is not hard to check that local consistency of these sets can be tested by B. It should be noted that the computation of U_x uses D_x to handle subpaths that drop the pebble outside of the subtree rooted at x and lift it sometime later.

Whether a path of type (2b) exists from node x can be inferred from U_x and the transitions $\delta_{q,x}$. Likewise, paths of type (2a) can be tested with the help of the sets $U_{x,y}$.

Thus, it remains to describe how to rule out paths of types (1a) and (1b) and how to check that the sets D_x are correct.

To this end, we define for every path π in T an increasing sequence $l_0, l_1, \ldots$ of nodes of π as follows. First of all, $l_0 = \varepsilon$. Given l_k, l_{k+1} is the minimal node $l > l_k$ on π such that the following conditions hold.

(i) For each state $q \in S_{l_k}$, each subpath of r starting from a node with configuration $(q, l_k, \bot)$ and reaching a configuration $(p, l, \bot)$ contains an accepting state.

---

[2] Since r is homogeneous, all these paths are isomorphic.

(ii) For each $y \not\succ l_k$ and each state $q \in S_{l_k, y}$, each subpath of r starting from a node with configuration (q, l_k, y), reaching a configuration (p, l, y) without lifting the pebble contains an accepting state.

(iii) For each $(p, q, \exists) \in D_{l_k}$, there is a path in r from (p, l_k, ε) to (q, l_k, ε) on which no node is a descendant of l.

(iv) For each $(p, q, +) \in D_{l_k}$, all paths in r from (p, l_k, ε) to (q, l_k, ε) contain an accepting state and do not visit any descendant of l.

With the help of König's Lemma, it is not hard to see that such an l exists if r is accepting.

For each k and each x with $l_k < x \leq l_{k+1}$, let R_x be the set of states q such that there is a node of r labeled with configuration $(q, x, \bot)$ that can be reached from a configuration $(p, l_k, \bot)$ for some $p \in S_{l_k}$, without passing any accepting state and without visiting any descendant of x. Note that $R_{l_{k+1}} = \emptyset$ by the definition of l_{k+1}.

Likewise, for each $x, y, y \not\succ x$, let $R_{x,y}$ be the set of states q such that there is a node of r with configuration (q, i, j) that can be reached from a configuration (q', l_k, j) to w , without passing any accepting state, without lifting the pebble, and without visiting descendants of x. Again by the definition of l_{k+1}, $U_{l_{k+1}} = \emptyset$.

Finally, let D'_x be a set of tuples $(p, q, \exists)$ and $(p, q, +)$ from D_i, which still have to be fulfilled in order to satisfy conditions (iii) and (iv) for k.

The accepting states of B are those for which $R_x = \emptyset$, for all $(S, U, R) \in X_x$, $R = \emptyset$, and $D'_x = \emptyset$.

It is not hard to see that B can maintain the characteristic vectors C_x and that B accepts T if and only if A has a homogeneous accepting run on T.

Furthermore, there are at most doubly exponentially many different possible sets X_x and thus the number of possible states of B is at most doubly exponential in the size of Q_A. Using standard space saving techniques, B can be constructed in space $|\Sigma_A| \cdot 2^{O(|Q_A|)}$. □

6 Conclusion

We have shown how to extend branching-time logics with hybrid machinery without blowing up complexity non-elementarily. The key to this result was the restriction to a single state variable proposed in [18].

We studied the satisfiability problem for the hybrid versions of several branching-time logics, ranging from $\mathcal{UB}$ to $\mathcal{ECTL}$ +Past. We proved **2EXPTIME**-completeness of the satisfiability problem in all cases. The lower bound was additionally explained by the succinctness of hybrid branching-time logics.

To abtain the upper complexity bound, we extended the automata-theoretic approach to hybrid branching-time logics: We proved non-emptiness of alternating one-pebble Büchi tree automata to be **2EXPTIME**-complete.

We want to give some open problems and directions for further research.

- There are a lot of open problems concerning the expressive power of hybrid branching-time logics. E.g., is $\mathcal{HB}(\mathbf{X}, \mathbf{F})$ a strict fragment of $\mathcal{HB}(\mathbf{X}, \mathbf{U})$?

- We only considered satisfiability, leaving out the model-checking problem. This gap has to be filled in future work.
- We restricted to $\mathcal{CTL}$-like branching-time logics, not allowing Boolean combinations and nesting of temporal operators inside a path-quantifier. Extending our results to such logics is a challenging problem. In particular, the complexity and expressiveness of hybrid $\mathcal{CTL}^*$ should be investigated.
- On the purely automata-theoretic side, the result on one-pebble tree automata should be extended to k-pebble tree automata.

References

1. M. Adler and N. Immerman. An $n!$ lower bound on formula size. *ACM TOCL*, 4(3):296–314, 2003.
2. C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In *Proc. of 13th CSL*, volume 1683 of *LNCS*, pages 307–321. Springer, 1999.
3. C. Areces, P. Blackburn, and M. Marx. Hybrid logics: Characterization, interpolation and complexity. *Journal of Symbolic Logic*, 66(3):977–1010, 2001.
4. C. Areces and B. ten Cate. Hybrid logics. In *Handbook of Modal Logic*, volume 3 of *Studies in Logic*, pages 821–868. Elsevier, 2007.
5. M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Informatica*, 20:207–226, 1983.
6. B. S. Chlebus. Domino-tiling games. *J. Comput. Syst. Sci.*, 32(3):374–392, 1986.
7. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proc. Logic of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.
8. E. A. Emerson and J. Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.
9. E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. of 32nd IEEE FOCS*, pages 368–377. IEEE, 1991.
10. M. Franceschet and M. de Rijke. Model checking hybrid logics (with an application to semistructured data). *Journal of Applied Logic*, 4(3):279–304, 2006.
11. M. Franceschet, M. de Rijke, and B.-H. Schlingloff. Hybrid logics on linear structures: Expressivity and complexity. In *Proc. of 10th TIME / 4th ICTL*, pages 192–202. IEEE, 2003.
12. V. Goranko. Temporal logic with reference pointers. In *Temporal logic*, volume 827 of *LNCS*, pages 133–148. Springer, 1994.
13. O. Kupferman and M. Y. Vardi. Memoryful branching-time logic. In *Proc. of 21st LICS*, pages 265–274. IEEE, 2006.
14. F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *Proc. of 17th LICS*, pages 383–392. IEEE, 2002.
15. F. Laroussinie and P. Schnoebelen. A hierarchy of temporal logics with past. *Theor. Comput. Sci.*, 148(2):303–324, 1995.
16. M. Mundhenk, T. Schneider, T. Schwentick, and V. Weber. Complexity of hybrid logics over transitive frames. In *Proc. of M4M-4*, volume 194 of *Informatik-Berichte*, pages 62–78. Humbold-Universität Berlin, 2005.
17. M. Rabin. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, pages 1–23. North Holland, 1970.
18. T. Schwentick and V. Weber. Bounded-variable fragments of hybrid logics. In *Proc. of the 24th STACS*, volume 4393 of *LNCS*, pages 561–572. Springer, 2007.

19. L. J. Stockmeyer. *The complexity of decision problems in automata theory and logic*. PhD thesis, MIT, 1974.

20. B. ten Cate and M. Franceschet. On the complexity of hybrid logics with binders. In *Proc. of 19th CSL*, volume 3634 of *LNCS*, pages 339–354. Springer, 2005.

21. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Sematics*, pages 133–192. Elsevier, MIT Press, 1990.

22. M. Y. Vardi. Alternating automata and program verification. In *Computer Science Today*, volume 1000 of *LNCS*, pages 471–485. Springer, 1995.

23. M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. of ICALP'98*, volume 1443 of *LNCS*, pages 628–641. Springer, 1998.

24. M. Y. Vardi. Automata-theoretic techniques for temporal reasoning. In *Handbook of Modal Logic*, volume 3 of *Studies in Logic*, pages 971–989. Elsevier, 2007.

25. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata and infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.