Memoryfull Branching-Time Logic

Orna Kupferman[1] and Moshe Y. Vardi[2]

[1] Hebrew University, School of Engineering and Computer Science, Jerusalem 91904, Israel
Email: orna@cs.huji.ac.il, URL: http://www.cs.huji.ac.il/~orna
[2] Rice University, Department of Computer Science, Houston, TX 77251-1892, U.S.A.
Email:vardi@cs.rice.edu, URL: http://www.cs.rice.edu/~vardi

Abstract. Traditional branching-time logics such as CTL* are memoryless: once a path in the computation tree is quantified at a given node, the computation that led to that node is forgotten. Recent work in planing suggests that CTL* cannot easily express temporal goals that refer to whole computations. Such goals require memoryfull quantification of paths. With such a memoryfull quantification, $E\psi$ holds at a node s of a computation tree if there is a path π starting at the root of the tree and going through s such that π satisfies the linear-time formula ψ. In this work we define the memoryfull branching-time logic mCTL* and study its expressive power and algorithmic properties. In particular, we show that while the satisfiability problem for mCTL* is 2EXPTIME-complete — not harder than that of CTL*, its model-checking problem is EXPSPACE-complete — exponentially harder than that of CTL*. The upper bounds are obtained by extending the automata-theoretic approach to handle memoryfull quantification, and are much more efficient than these obtained by translating mCTL* to branching logics with past.

1 Introduction

Since the introduction of temporal logic into computer science by Pnueli in [Pnu77], the family of temporal logics has been widely accepted as an appropriate formal framework for the description of dynamic behavior, cf. [MP92]. Within this family, the branching-time logic CTL* has emerged as one of the more expressive logics [EH86], unifying the linear-time logic LTL [Pnu77] and the branching-time logic CTL [CE81]. While the modal fixpoint logic [Koz83] is more expressive than CTL*, it is in some sense a low-level logic, making it an "unfriendly" logic for users, whereas CTL* can naturally express complex properties of computation trees. While the appropriateness of a branching-time logic for practical formal verification is debatable [Var01], CTL* is the right logic for applications that require computation-tree reasoning.

An example of an application that requires computation-tree reasoning is sanity checks for the modeling of a system: verification is done with respect to a mathematical model of the system, and it is important to check that the model, whose generation typically involves abstraction and reduction techniques, is correct. For example, in order to detect vacuous satisfaction of specifications, one has to check the satisfaction of *witness formulas* [KV03], which are existential formulas describing a nontrivial behavior of the model. Likewise, in order to check that the model does not disable essential behaviors of the system, one has to check *possibility properties* [Lam98], which require computations of the model to be extendible to computations exhibiting the behaviors.

Another example of a setting in which computation-tree reasoning is required is *automated task planning* [FN71,PW92], where given a description of a dynamic domain and of the basic actions that can be performed on it, and given a goal that defines a success condition to be achieved, one has to find a suitable plan, that is, a description of the actions to be executed on the domain in order to achieve the goal. "Classical" planning concentrates on the so called "reachability" goals, that is, on goals that define a set of final desired states to be reached (e.g., the red block is above the yellow block). Quite often, practical applications require plans that deal with goals that are more general than sets of final states. Several planning approaches have been recently proposed, where *temporal logic* formulas are used as goal languages, thus allowing for goals that define conditions on the whole plan execution paths, i.e., on the sequences of states resulting from the execution of plans (see, e.g., [BK98,BK00,**?**,CM98,DLPT02,dGV99,KD01,PT01]). Most of these approaches use LTL as the goal language. The temporal logic LTL allows one to express reachability goals (e.g., Fq — reach q), maintainability goals (e.g., Gq — maintain q), as well as goals that combine reachability and maintainability requirements (e.g., FGq — reach a set of states where q can be maintained), and Boolean combinations of these goals.

In *planning in nondeterministic domains* [CRT98b,PS92,War76], actions are allowed to have different outcomes, and it is not possible to know at planning time which of the different possible outcomes will actually take place. Nondeterminism in action outcome is necessary for modeling in a realistic way several practical domains (e.g., robotics, autonomous controllers, etc.). For instance, in a realistic robotic application one has to take into account that actions like "pick up object" might result in a failure (e.g., if the object slips out of the robot's hand). A consequence of nondeterminism is that the execution of a plan may lead to more than one possible execution path. Therefore, one has to distinguish between the case the given goal has to be satisfied by all the possible execution paths ("strong" planning) and the case where the goal has to be satisfied only by some of the possible execution paths ("weak" planning). In the case of an LTL goal φ, strong planning corresponds to interpreting the formula in a universal way, as the CTL* formula as $A\varphi$, while weak planning corresponds to interpret it in an existential way, as the CTL* formula $E\varphi$.

Weak and strong plans are two very extreme ways of satisfiability of an LTL formula. In practical applications, it might be impossible to achieve goals in a strong way: for instance, in the robotic application it might be impossible to fulfill a given task if objects keep slipping from the robot's hand. On the other hand, weak plans are too unreliable, since they achieve the goal only under overly optimistic assumptions on the outcomes of action executions. In the case of reachability goals, *strong cyclic planning* [CRT98a,**?**] has been shown to provide a viable compromise between weak and strong planning. Formally, a plan is strong cyclic if each possible partial execution of the plan can always be extended to an execution that reaches some goal state. Strong cyclic planning allows for plans that encode iterative trial-and-error strategies, like "pick up an object until succeed". The execution of such strategies may loop forever only in the case the action "pick up object" continuously fails, and a failure in achieving the goal in the case of such an unfair execution is usually acceptable. This "always possibly" approach corresponds to Lamport's possibility properties mentioned above.

Inspired by the work on strong cyclic planning, Pistore and Vardi went on to explore the different degrees in which an LTL formula φ can be satisfied that exist between the strong goal $A\varphi$ and the weak goal $E\varphi$ [PV03]. They showed that a two-player game can

model the spectrum between strong and weak planning. Player A chooses action outcomes in order to make goal φ fail, while player E chooses action outcomes in order to satisfy the goal φ. The degree of strength of the temporal requirement is determined by the structure of the game: who makes the first move and how many turns. Pistore and Vardi then proposed a logic based on these path games, using path quantifiers $\mathcal{A}$ and $\mathcal{E}$, which can express the spectrum between strong and weak planning.

While the Pistore-Vardi logic is a branching-time logic, quite close to CTL*, it is different from CTL*. In CTL*, a path quantifier turns a path formula into a pure state formula. Thus, for example, the formula $AGE\varphi$ holds at a start state s_0 if for every state s reachable from s_0 there is a path starting at s that satisfies φ; the truth of $E\varphi$ depends only on the state in which it is evaluated. In contrast, in the Pistore-Vardi logic the formula $\mathcal{AE}\varphi$ holds at a start state s_0 if for every state s reachable from s_0 there is a path starting at s_0 and continuing through s that satisfies φ; the truth of $\mathcal{E}\varphi$ in a state depends also on the path that lead to that state. In other words, while CTL* path quantifiers are *memoryless*, the path quantifiers $\mathcal{A}$ and $\mathcal{E}$ are *memoryfull*[1]. The Pistore-Vardi logic, however, does not have a standard logical syntax; for example, it is not closed under conjunction and disjunction.

We propose in this paper a memoryfull variant of CTL*, which unifies CTL* and the Pistore-Vardi logic. We name the new logic mCTL*. Semantically, mCTL* is obtained from CTL* by reinterpreting the path quantifiers of the logic to be memoryfull[2]. With memoryfull quantification, $E\varphi$ holds at a node s of a computation tree if there is a path π starting at the root of the tree and going through s such that π satisfies the linear-time formula φ. Syntactically, we add a special proposition $present$, which is needed to emulate the ability of CTL* to talk about the "present". One of our main results is that the addition of $present$ is needed to make mCTL* at least as expressive as CTL*. (As in [BGK03], one can translate mCTL* to CTL* via path logic, but this translation is of nonelementary complexity. We note that $present$ is not required to easily express the Pistore-Vardi logic in mCTL*; the strong cyclic goal of $\mathcal{AE}\varphi$ is expressed in mCTL* by the formula $AGE\varphi$.) We show that memoryfull quantification can be expressed in CTL* extended with the ability to refer to the *linear past* [KP95]. We note that the proposition $present$ of mCTL* is different from the "*Now*" used in [FLS02], in the context of branching temporal logics with past. While the "*Now*" in [FLS02] is a unary temporal operator, which chops away the past, our $present$ is an atomic proposition that holds in the present.

We then examine two decision problems related to mCTL*: satisfiability and model checking. We first show that satisfiability of mCTL*is 2EXPTIME-complete, which is the same complexity as that of CTL* [EJ88,VS85]. Establishing the upper bound is quite nontrivial, as the standard automata-theoretic approach to CTL* satisfiability [ES84,KVW00] is strongly based on the memoryless character of CTL* quantifiers. We describe an extension of the automata-theoretic framework that can handle memoryfull path quantifiers. We then show that model checking mCTL*is significantly harder than model checking CTL*. Model checking mCTL*is EXPSPACE-complete, while model checking CTL* is PSPACE-complete [EL85]. Since our lower-bound proof uses an mCTL*- formula corresponding to Lamport's possibility properties, it also settles an open question on the complexity of model checking of such properties, as well as the Pistore-Vardi logic. In

[1] It is shown in [BGK03] that the Pistore-Vardi logic is expressible in CTL*, but only a nonelementary translation, which goes through *path logic* [HT87], is known.

[2] Strictly speaking, mCTL*covers only the finite-alternation fragment of the Pistore-Vardi logic, leaving out the infinite-alternation fragment.

addition, our results, together with the linear translation of mCTL* to CTL* with linear past, imply new results about the complexity of branching temporal logic with past [LS00,FLS02,Mar03]. Our results show that the transition from memoryless to memoryfull quantifiers is significant. We can see now that the significant complexity gap between CTL* satisfiability, which is 2EXPTIME-complete, and CTL* model checking, which is PSPACE-complete, is due to quantifier memoryless, which helps model checking [EL85] but not satisfiability [VS85].

2 Preliminaries

2.1 The temporal logic memoryfull CTL*

The branching-time logic *memoryfull CTL** (mCTL*, for short) combines both branching-time and linear-time operators [EH86]. A path quantifier E ("for some path") can prefix an assertion composed of an arbitrary combination of the linear-time operators X ("next time") and U ("until"). The syntax of mCTL* is similar to the syntax of CTL*: there are two types of formulas in mCTL*: *state formulas*, whose satisfaction is related to a specific state, and *path formulas*, whose satisfaction is related to a specific path. The only difference in the syntax between the two logics is the fact mCTL* formulas may refer to a special atomic proposition $present$, which holds only in the present. Formally, let AP be a set of atomic proposition names. An mCTL* state formula is either:

- **true**, **false**, $present$, or p, for all $p \in AP$;
- $\neg\varphi_1$ or $\varphi_1 \vee \varphi_2$, where φ_1 and φ_2 are mCTL* state formulas;
- $E\psi$, where ψ is an mCTL* path formula.

An mCTL* path formula is either:

- An mCTL* state formula;
- $\neg\psi_1$, $\psi_1 \vee \psi_2$, $X\psi_1$, or $\psi_1 U \psi_2$, where ψ_1 and ψ_2 are mCTL* path formulas.

The logic mCTL* consists of the set of state formulas generated by the above rules.

We define the *size* $|\varphi|$ of φ as the number of state and path subformulas that φ has. Note that our definition of size corresponds to the number of nodes in a reduced DAG representation of the formula. We use the usual $\wedge$ and $\rightarrow$ Boolean abbreviations. Additional temporal operators can be obtained from X, U, and their negations (e.g., $F\psi_1 = \mathbf{true}U\psi_1$ and $G\psi_1 = \neg F\neg\psi_1$). Finally, the path quantifier A ("for all paths") can be obtained by negating the path quantifier E (i.e., $A\psi = \neg E\neg\psi$).

The semantics of mCTL* is defined with respect to *computation trees*. Given a finite set D of *directions*, a *D-tree* is a set $T \subseteq D^*$ such that if $x \cdot d \in T$ where $x \in D^*$ and $d \in D$, then also $x \in T$. The elements of T are called *nodes*, and the empty word ε is the *root* of T. The prefix relation induces a partial order $\leq$ between nodes of T. Thus, for two nodes x and y, we say that $x \leq y$ iff there is some $z \in D^*$ such that $y = x \cdot z$. For every $x \in T$, the nodes $x \cdot d$, for $d \in D$, are the *successors* of x. The *direction* of a node $x \cdot d$ is d. The direction of the root is some designated $d \in D$, referred to as the *root direction*. We denote the direction of a node $y \in T$ by $dir(y)$. A node is a *leaf* if it has no successors. For a node $x \in T$, an *x-path* of T is a minimal set $\pi \subseteq T$ such that $x \in \pi$ and for every $y \in \pi$, either y is a leaf or there exists a unique $d \in D$ such that $y \cdot d \in \pi$. When $x = \varepsilon$,

we say that π is a *path*. For a path π and a node $x \in \pi$, the *suffix* of π that starts at x is the x-path $\pi \cap \{y : x \leq y\}$.

Given an alphabet Σ, a *Σ-labeled D-tree* is a pair $\langle T, \tau \rangle$ where T is a D-tree and $\tau : T \to \Sigma$ maps each node of T to a letter in Σ. Of special interest to us are Σ-labeled trees in which $\Sigma = 2^{AP}$ for some set AP of atomic propositions. We call such Σ-labeled trees *computation trees*. A computation tree is often given by means of a *Kripke structure* $K = \langle AP, W, R, w_{in}, L \rangle$, where AP is the set of atomic propositions, W is a set of states, $R \subseteq W \times W$ is a transition relation that must be total (i.e., for every $w \in W$ there exists $w' \in W$ such that $\langle w, w' \rangle \in R$), w_{in} is an initial state, and $L : W \to 2^{AP}$ maps each state to the set of atomic propositions true in that state. A *path* in K is an infinite sequence of states, $\pi = w_0, w_1, \ldots$ such that $\langle w_i, w_{i+1} \rangle \in R$ for every $i \geq 0$. We define the size $\|K\|$ of K as $|W| + |R|$. The Kripke structure K induces a computation tree $\langle T_K, \tau_K \rangle$ that corresponds to the unwinding of K from w_{in}. Formally, $\langle T_K, \tau_K \rangle$ is a 2^{AP}-labeled W-tree where $T_K \subseteq W^*$ contains exactly all the prefixes of paths in K that start at w_{in} and $\tau_K(x)$, for $x \in T_K$, is $L(dir(x))$, with w_{in} being the root direction.

The difference between mCTL* and CTL* is in the semantics of path quantification. Consider a computation tree. In CTL*, path quantification ranges over paths that start in the current node. In mCTL*, path quantification ranges over paths that start at the root and visit the current node. For example, the CTL* formula $AGE\psi$, for a linear formula ψ holds in a root of a computation tree if a computation satisfying ψ starts from every node in the tree. When viewed as an mCTL* formula, it holds in a computation tree if for every node x of the tree, the path from the root to x can be extended to a path satisfying ψ. As discussed in Section 1, this corresponds to string cyclic plans [CRT98b,DTV99]. In particular, when $\varphi = Fp$, it states that p has either occurred in the past or is possible in the future. This corresponds to Lamport's possibility properties [Lam98]. Thus, when evaluating path formulas, one cannot ignore the past and satisfaction may depend on the events that have taken place since the beginning of the execution and until the present. Below we define the semantics of mCTL* formally.

Consider a computation tree $\langle T, \tau \rangle$. For two nodes x and c in T and an mCTL* formula φ, we use $x, c \models \varphi$ to indicate that x satisfies φ with c being the present. Similarly, $\pi, x, c \models \psi$, for a path formula ψ, iff the suffix of the path π that starts at x satisfies ψ with c being the present. Formally, we have the following (for known T and τ).

- For all x, c in T, we have $x, c \models$ **true** and $x, c \not\models$ **false**.
- $x, c \models p$, for $p \in AP$, iff $p \in \tau(x)$.
- $x, c \models present$ iff $x = c$.
- $x, c \models \neg\varphi_1$ iff $x, c \not\models \varphi_1$.
- $x, c \models \varphi_1 \vee \varphi_2$ iff $x, c \models \varphi_1$ or $x, c \models \varphi_2$.
- $x, c \models E\psi$ iff there exists a path π such that $x \in \pi$ and $\pi, \varepsilon, x \models \psi$.
- $\pi, x, c \models \varphi$ for a state formula φ, iff $x, c \models \varphi$.
- $\pi, x, c \models \neg\psi_1$ iff $\pi, x, c \not\models \psi_1$.
- $\pi, x, c \models \psi_1 \vee \psi_2$ iff $\pi, x, c \models \psi_1$ or $\pi, x, c \models \psi_2$.
- $\pi, x, c \models X\psi$ iff $\pi, x \cdot d, c \models \psi$, where $d \in D$ is such that $x \cdot d \in \pi$.
- $\pi, x, c \models \psi_1 U \psi_2$ iff π contains a node $y \geq x$ such that $\pi, y, c \models \psi_2$ and for all $x \leq z < y$, we have $\pi, z, c \models \psi_1$.

We say that a computation tree $\langle T, \tau \rangle$ satisfies an mCTL* formula φ iff $\varepsilon, \varepsilon \models \varphi$. Note that while in CTL* path quantification ranges over paths that start in the current node, here

path quantification ranges over paths that start at the root and visit the current node. Thus, when evaluating path formulas, one cannot ignore the past and satisfaction may depend on the events that have taken place since the beginning of the execution and until the present. Note also that formulas of the form $E\psi$ "reset the present"; thus the satisfaction of $E\psi$ with respect to x, c is independent of c, and the present is set to x.

The logic mCTL*- is a fragment of mCTL* in which the atomic proposition $present$ is not allowed. Thus, while CTL* and mCTL* formulas can directly refer to the present (in CTL*, path quantification starts in the present, and in mCTL*, $present$ holds only in the present), this is not the case for mCTL*- formulas. As we show in Section 3, mCTL*- cannot emulate the ability of CTL* to refer to the present, and is strictly weaker than mCTL*.

Examples Consider the mCTL* formula $AG(p \rightarrow EFq)$. The formula states that whenever a node x is labeled with p, then some path that goes through x has a node labeled q. Thus, whenever p holds, it is linearly ordered with q. Note that the specification of this property in CTL* is much more complicated.

The formula $AG(grant \rightarrow EF(req \wedge F(ack \wedge F present)))$ demonstrates how mCTL* formulas can use the ability to refer to the present with $present$ in order to refer to the past. Indeed, the formula states that whenever a node x is labeled with $grant$, the path from the root to x has a node labeled with req, followed by a node labeled with ack, followed by the present. Thus, grants are given only if a request was issued and then acknowledged in the past.

Finally, The formula $AG(ack \rightarrow EF(req \wedge F(present \wedge F grant)))$ gives a different view of the same sequence of events and demonstrates how mCTL* formulas can refer to both the past and the future. Indeed, the formula states that whenever a node x is labeled with ack, the path from the root to x has a node labeled with req and some path that starts in x has a node labeled with $grant$. Thus, acknowledgments are given only if a request was issued in the past and and is going to be granted in the future. □

The definition of mCTL* leads to natural theoretical and practical questions: in the theoretical front, we would like to study how the transition to a memoryfull path quantification influences the expressive power of the logic, its connection to logics with past, and to standard CTL*, as well as the necessity of the atomic proposition $present$. In the practical front, we would like to study how the transition influences the complexity of the satisfiability and the model-checking problems. The later is of particular interest as model-checking algorithms for CTL* are based on a bottom-up reasoning, where internal state formulas are evaluated first and replaced by new atomic propositions [EL85]. Such reasoning cannot be applied with a memoryfull path quantification.

3 Expressiveness

In this section we discuss the expressive power of mCTL* and mCTL*-. We show that while mCTL* and CTL* are equally expressive, the ability to point to the present is crucial, thus mCTL*- is strictly weaker than mCTL*.

Theorem 1. *mCTL* and CTL* are equally expressive.*

Proof: We first prove that every CTL* formula has an equivalent mCTL* formula of linear length. Consider a CTL* formula ψ. An equivalent mCTL* formula can be obtained from ψ by replacing each subformula of the form $E\xi$ by the formula $EF(present \wedge \xi)$. Clearly, the translation is linear. For the other direction, we show that every mCTL* formula can be translated to a monadic SnS formula in which all sets quantifiers are over paths. By [HT87], this fragment of SnS is as expressive as CTL*. Since reference to the past is easy in first-order logic, the translation of mCTL* formulas to monadic SnS formula in which all sets quantifiers are over paths is similar to the one described in [HT87] for CTL*. □

Note that while the blow up in translating a CTL* formula to an mCTL* formula is linear, the translation in the other direction goes via monadic SnS and is of nonelementary complexity[3]. We now show that this complication originates from the ability of mCTL* to refer to the past. Thus, if we extend CTL* with past temporal operators, the translation is linear. Let CTL$^\star_{lp}$ and CTL$^\star_{lp}$- denote the extension of CTL* with past operators, with and without $present$, respectively, where formulas are interpreted over computation trees, thus past is linear [KP95].

Theorem 2. *Every mCTL* (mCTL*-) formula has an equivalent CTL$^\star_{lp}$ (resp. CTL$^\star_{lp}$-) formula of linear length.*

Proof: Consider an mCTL* (mCTL*-) formula ψ. An equivalent CTL$^\star_{lp}$ (resp. CTL$^\star_{lp}$-) formula can be obtained from ψ by replacing each subformula of the form $E\xi$ by the formula $E\overline{F}(init \wedge \xi)$, where $\overline{F}$ is the past counterpart o of F (that is, $\overline{F}\xi$ iff ξ holds along some suffix that starts in the past), and $init$ is a special atomic proposition that labels the root of the tree (alternatively, $init$ can be replaced by $A\overline{X}$**false**, which holds only at the root). Clearly, the translation is linear. □

We note that the complexity of the model-checking and the satisfiability problems for CTL$^\star_{lp}$ and CTL$^\star_{lp}$- is open [KP95], thus Theorem 2 does not lead to a model-checking or decidability procedure. In fact, the translation, together with the lower bounds we describe in Section 4 for mCTL* and mCTL*improves the known lower bounds known for CTL$^\star_{lp}$ and CTL$^\star_{lp}$-.

Recall that the proposition $present$ emulates the ability of CTL* to talk about the present. Indeed, the translation we described, of CTL* formulas to mCTL* formulas, uses $present$. We now show that the use of $present$ is essential, thus mCTL*- is strictly weaker than mCTL* (and thus, also CTL*).

Theorem 3. *There is a CTL formula with no equivalent mCTL*- formula.*

Proof: We prove that the CTL formula $\psi = EF(EXp \wedge EX\neg p)$ has no equivalent mCTL*- formula.

For $n \geq 0$, let K_n and K'_n be the Kripke structures described in Figure 1. It is not hard to see that K_n satisfies ψ and K'_n does not. We prove that for all $n \geq 0$, no mCTL*-formula φ with less than n X operators can distinguish between K_n and K'_n. Note that this implies that no mCTL*- formula is equivalent to ψ. Indeed, if we assume by way of contradiction that such a formula φ exists, it has some fixed number n of X operators, so it cannot distinguish between M_{n+1} and M'_{n+1}, whereas ψ does distinguish between them.

[3] In Section 4 we show that mCTL* is at least exponentially more succinct than CTL*

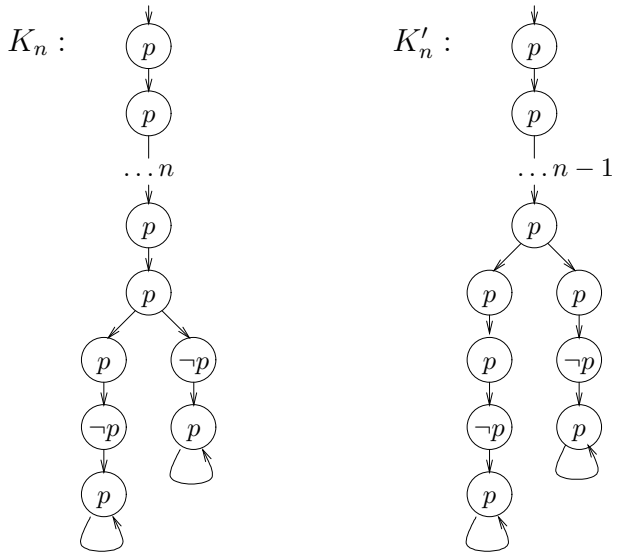

Fig. 1. The Kripke structures K_n and K'_n.

The proof is based on Wolper's result that no LTL formula with less than n X operators, for $n \geq 0$, can distinguish between $p^n \cdot \neg p \cdot p^\omega$ and $p^{n+1} \cdot p \cdot p^\omega$ [Wol81]. We extend Wolper's result to computations and formulas over several atomic propositions. For a computation π and an atomic proposition p, we denote by $\pi_{|p}$ the projection of π on p. For two paths π and ρ, an atomic proposition p, and $n \geq 0$, we say that π and ρ are (p, n)-*indistinguishable* if one of the following holds:

- $\pi_{|p} = \rho_{|p} = p^\omega$,
- $\pi_{|p} = \rho_{|p} = (\neg p)^\omega$,
- $\pi_{|p} = p^n \cdot \neg p \cdot p^\omega$ and $\rho_{|p} = p^{n+1} \cdot \neg p \cdot p^\omega$, or
- $\pi_{|p} = (\neg p)^n \cdot p \cdot (\neg p)^\omega$ and $\rho_{|p} = (\neg p)^{n+1} \cdot p \cdot (\neg p)^\omega$.

Lemma 1. *Consider an LTL formula ξ over the set AP of atomic propositions. Let π and ρ be (p, n)-indistinguishable for all $p \in AP$. If ξ has less than n X operators, then ξ cannot distinguish between π and ρ.*

The proof of Lemma 1 is by induction on the structure of ξ and is similar to the proof in [Wol81], which corresponds to the special case where $|AP| = 1$.

Let $\langle T_n, \tau_n \rangle$ and $\langle T'_n, \tau'_n \rangle$ be the $2^{\{p\}}$-labeled $\{0, 1\}$-trees corresponding to the unwinding of K_n and K'_n, respectively. Let $P_n \subseteq T_n$ be such that $x \in P_n$ iff $\tau_n(x) = \{p\}$, and let $Q_n \subseteq T_n$ be such that $x \in Q_n$ iff $\tau_n(x) = \emptyset$. Define P'_n and Q'_n similarly with respect to $\langle T'_n, \tau'_n \rangle$. Note that $|Q_n| = |Q'_n| = 2$.

Consider an mCTL*- formula φ. For a state subformula θ of φ (that is, θ is an atomic proposition, a formula of the form $E\xi$, or a Boolean combination of them), denote by $f(\theta)$ and $f'(\theta)$ the set of nodes that satisfy θ in $\langle T_n, \tau_n \rangle$ and $\langle T'_n, \tau'_n \rangle$, respectively. We claim that all the nodes in P_n agree about the satisfaction of θ, and similarly for P'_n, Q_n, and

Q'_n. Moreover, θ is satisfied in (all the nodes in) P_n iff it is satisfied in (all the nodes in) P'_n, and similarly for Q_n and Q'_n. Formally, we have the following.

Lemma 2. *For every state subformula θ of φ, the following holds:*

1. $P_n \subseteq f(\theta)$ *or* $P_n \cap f(\theta) = \emptyset$,
2. $Q_n \subseteq f(\theta)$ *or* $Q_n \cap f(\theta) = \emptyset$,
3. $P'_n \subseteq f(\theta)$ *or* $P'_n \cap f(\theta) = \emptyset$,
4. $Q'_n \subseteq f(\theta)$ *or* $Q'_n \cap f(\theta) = \emptyset$,
5. $P_n \subseteq f(\theta)$ *iff* $P'_n \subseteq f'(\theta)$, *and*
6. $Q_n \subseteq f(\theta)$ *iff* $Q'_n \subseteq f'(\theta)$.

Proof: The proof proceeds by an induction on the structure of θ. If θ is an atomic proposition, then $\theta = p$, so $f(\theta) = P_n$, $f'(\theta) = P'_n$, and we are done. If θ is a Boolean combination of state formulas, the lemma follows from the induction hypothesis. If θ is of the form $E\xi$, we proceed by an internal induction on the nesting depth of existential path quantifiers in ξ. For the induction base, if there is no nested path quantifier in ξ, then ξ is an LTL formula over p. Therefore, by Lemma 1, the formula ξ cannot distinguish between $p^n \cdot \neg p \cdot p^\omega$ and $p^{n+1} \cdot \neg p \cdot p^\omega$ — the labels of the two paths of both $\langle T_n, \tau_n \rangle$ and $\langle T'_n, \tau'_n \rangle$. Therefore, for every node x in T_n or T'_n, the value of $E\xi$ in x is the (same) value of ξ in these paths, and we are done. For the induction step, let $\theta_1, \ldots, \theta_k$ be the maximal strict state subformulas of ξ. Let $q_1, \ldots, q_k$ be new atomic propositions corresponding to $\theta_1, \ldots, \theta_k$. Consider an extension of τ and τ' to $2^{\{p, q_1, \ldots, q_k\}}$ such that for every $1 \leq i \leq k$ and $x \in T_n$, we have that $q_i \in \tau(x)$ iff x satisfies θ_i. Similarly, for every $x \in T'_n$, we have that $q_i \in \tau'(x)$ iff x' satisfies θ_i. The induction hypothesis guarantees that the paths of $\langle T_n, \tau_n \rangle$ and $\langle T'_n, \tau'_n \rangle$ are pairwise (q_i, n)-indistinguishable, for all $1 \leq i \leq k$. Hence, by Lemma 1, the formula ξ cannot distinguish between the paths of $\langle T_n, \tau_n \rangle$ and $\langle T'_n, \tau'_n \rangle$. Therefore, for every node x in T_n or T'_n, the value of $E\xi$ in x is the (same) value of ξ in these paths, and we are done. □

Lemma 2 implies that for every mCTL*- formula φ with less than n X operators, φ is satisfied in the root of $\langle T_n, \tau_n \rangle$ iff φ is satisfied in the root of $\langle T'_n, \tau'_n \rangle$. Thus, no mCTL*-formula with less than n X's can distinguish between K_n and K'_n, and we are done. □

The fact that mCTL*- loses track of the present whenever path quantification is applied weakens its branching nature. This is reflected in the fact that the CTL formula we have used in the proof of Theorem 3 is similar to the formula with which Milner shows the differences between trace equivalence and bisimulation [Mil80]. This weakness of mCTL*-motivates us to focus, in the rest of this paper, in the logic mCTL*.

4 Decision Procedures

In this section we study the model-checking and satisfiability problems for mCTL*. We show that while the satisfiability problem is not harder than the one for CTL*, the model-checking problem is exponentially harder. For the upper bounds, we need to develop an automata-theoretic approach for memoryfull branching temporal logic. For that, we start with definitions of alternating automata and introduce *alternating automata with satellites*. We then translate mCTL* formulas to such automata.

4.1 Alternating automata

Automata over infinite trees (tree automata) run over Σ-labeled D-trees that have no leaves [Tho90]. *Alternating tree automata* generalize nondeterministic tree automata and were first introduced in [MS87]. Here we define *symmetric* alternating automata, which cannot distinguish between the different successors of a node, and send copies to the successors in either a universal or an existential manner [JW95,Wil99].

Let $\Omega = \{\Box, \Diamond\}$, and let $\mathcal{B}^+(\Omega \times Q)$ be the set of positive Boolean formulas over $\Omega \times Q$; i.e., Boolean formulas built from elements in $\Omega \times Q$ using $\wedge$ and $\vee$, where we also allow the formulas **true** and **false** and, as usual, $\wedge$ has precedence over $\vee$. For a set $S \subseteq \Omega \times Q$ and a formula $\theta \in \mathcal{B}^+(\Omega \times Q)$, we say that S *satisfies* θ iff assigning **true** to elements in S and assigning **false** to elements in $(\Omega \times Q) \setminus S$ makes θ true.

Consider a set D of directions. In a nondeterministic automaton $\mathcal{A}$ over Σ-labeled D-trees, with a set Q of states, the transition function δ maps an automaton state $q \in Q$ and an input letter $\sigma \in \Sigma$ to a set of tuples of states. Each tuple suggests a nondeterministic choice for the automaton's next configuration. When the automaton is in a state q and is reading a node x with successors $x \cdot d_1, \ldots, x \cdot d_n$, and labeled by a letter σ, it proceeds by first choosing a tuple $\langle q_1, \ldots, q_n \rangle \in \delta(q, \sigma)$ and then splitting into n copies, where copy i enters the state q_i and proceeds to the node $x \cdot d_i$. In a symmetric automaton, the transition function δ maps q and σ to a formula in $\mathcal{B}^+(\Omega \times Q)$. Intuitively, an atom $\langle \Box, q \rangle$ corresponds to copies of the automaton in state q, sent to all the successors of the current node. An atom $\langle \Diamond, q \rangle$ corresponds to a copy of the automaton in state q, sent to some successor of the current node. When, for instance, the automaton is in state q, reads a node x with successors $x \cdot d_1, \ldots, x \cdot d_n$, and $\delta(q, V(x)) = (\Box, q_1) \wedge (\Diamond, q_2) \vee (\Diamond, q_2) \wedge (\Diamond, q_3)$, the automaton can either send n copies in state q_1 to all the successors of x and a copy in state q_2 to one of the successors, or send a copy in state q_2 to one of the successors and a copy in state q_3 to one (possibly the same) successor. So, while nondeterministic tree automata send exactly one copy to each successor, symmetric automata can send several copies to the same successor. On the other hand, symmetric automata cannot distinguish between left and right and can send copies to successor nodes only in either a universal or an existential manner.

Formally, a symmetric automaton is a tuple $\mathcal{A} = \langle \Sigma, Q, \delta, q_{in}, \alpha \rangle$ where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to \mathcal{B}^+(\Omega \times Q)$ is a transition function, $q_{in} \in Q$ is an initial state, and α specifies the acceptance condition (a condition that defines a subset of Q^ω). Let $T = D^*$. A *run* of a symmetric automaton $\mathcal{A}$ on an input Σ-labeled D-tree $\langle T, \tau \rangle$ is a tree $\langle T_r, r \rangle$ (to be formally defined shortly) in which each node is labeled by an element of $D^* \times Q$. Unlike T, in which each node has exactly $|D|$ children, the tree T_r may have nodes with many children and may also have leaves. Thus, $T_r \subset \mathbb{N}^*$ and a path in T_r may be either finite, in which case it ends in a leaf, or infinite. Each node of T_r corresponds to a node of T. A node in T_r, labeled by (x, q), describes a copy of the automaton that reads the node x of T and visits the state q. Note that many nodes of T_r can correspond to the same node of T; in contrast, in a run of a nondeterministic automaton on $\langle T, \tau \rangle$ there is a one-to-one correspondence between the nodes of the run and the nodes of the tree. The labels of a node and its children have to satisfy the transition function. Formally, the run $\langle T_r, r \rangle$ is a $(D^* \times Q)$-labeled $\mathbb{N}$-tree that satisfies the following:

1. $\varepsilon \in T_r$ and $r(\varepsilon) = (\varepsilon, q_{in})$.

2. Let $y \in T_r$ with $r(y) = (x, q)$ and $\delta(q, V(x)) = \theta$. Then there is a (possibly empty) set $S \subseteq \Omega \times Q$, such that S satisfies θ, and for all $(c, s) \in S$, the following hold:
   - If $c = \Box$, then for each $d \in D$, there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (x \cdot d, s)$.
   - If $c = \Diamond$, then for some $d \in D$, there is $j \in \mathbb{N}$ such that $y \cdot j \in T_r$ and $r(y \cdot j) = (x \cdot d, s)$.

For example, if $\langle T, \tau \rangle$ is a $\{1, 2\}$-tree with $\tau(\varepsilon) = a$ and $\delta(q_{in}, a) = \Diamond q_1 \wedge \Box q_2$, then the nodes of $\langle T_r, r \rangle$ at level 1 include one of the labels $(1, q_1)$ or $(2, q_1)$, and include both labels $(1, q_2)$ and $(2, q_2)$. Note that if $\theta =$ **true**, then y need not have children. This is the reason why T_r may have leaves. Also, since there exists no set S as required for $\theta =$ **false**, we cannot have a run that takes a transition with $\theta =$ **false**. Each infinite path ρ in $\langle T_r, r \rangle$ is labeled by a word in Q^ω. Let $inf(\rho)$ denote the set of states in Q that appear in $r(\rho)$ infinitely often. A run $\langle T_r, r \rangle$ is accepting iff all its infinite paths satisfy the acceptance condition. In *Büchi* automata, $\alpha \subseteq Q$, and an infinite path ρ satisfies α iff $inf(\rho) \cap \alpha \neq \emptyset$. In *co-Büchi* automata, $\alpha \subseteq Q$, and an infinite path ρ satisfies α iff $inf(\rho) \cap \alpha = \emptyset$.

An automaton accepts a tree iff there exists an accepting run on it. We denote by $\mathcal{L}(\mathcal{A})$ the language of the automaton $\mathcal{A}$; i.e., the set of all labeled trees that $\mathcal{A}$ accepts. We say that $\mathcal{A}$ is *nonempty* iff $\mathcal{L}(\mathcal{A}) \neq \emptyset$. We denote by $\mathcal{A}^q$ the automaton obtained from $\mathcal{A}$ by making q the initial state. The *complement* of an automaton $\mathcal{A}$ is an automaton $\tilde{A}$ that accepts exactly all the tress that $\mathcal{A}$ rejects.

In [KVW00], the authors introduce *hesitant alternating automata* (HAAs, for short) and show that CTL* formulas can be translated to such automata. An HAA is an alternating automaton $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, \alpha \rangle$, where $\alpha = \langle G, B \rangle$ with $G \subseteq Q$ and $B \subseteq Q$. That is, the acceptance condition of HAAs consists of a pair of sets of states. As in weak alternating automata [MSS88], there exists a partition of Q into disjoint sets $Q_1, \ldots, Q_m$ and a partial order $\leq$ on these sets such that transitions lead to sets that are lower in the partial order. Formally, for every $q \in Q_i$ and $q' \in Q_j$ for which q' occurs in $\delta(q, \sigma)$, for some $\sigma \in \Sigma$, we have $Q_j \leq Q_i$. In addition, each set Q_i is classified as either *transient*, *existential*, or *universal*, such that for each set Q_i and for all $q \in Q_i$ and $\sigma \in \Sigma$, the following hold:

1. If Q_i is a transient set, then $\delta(q, \sigma)$ contains no states of Q_i.
2. If Q_i is an existential set, then $\delta(q, \sigma)$ only contains disjunctively related states of Q_i. Thus, no state of Q_i is in a scope of a $\Box$, and if $\delta(q, \sigma)$ is rewritten in DNF, then there are no two states of Q_i in the same disjunct.
3. If Q_i is a universal set, then $\delta(q, \sigma)$ only contains conjunctively related elements of Q_i. Thus, no state of Q_i is in a scope of a $\Diamond$, and if $\delta(q, \sigma)$ is rewritten in CNF, then there are no two states of Q_i in the same conjunct.

It follows that every infinite path π of a run $\langle T_r, r \rangle$ gets trapped within some existential or universal set Q_i. The path then satisfies an acceptance condition $\langle G, B \rangle$ if and only if either Q_i is an existential set and $inf(\pi) \cap G \neq \emptyset$, or Q_i is a universal set and $inf(\pi) \cap B = \emptyset$. Note that the acceptance condition of HAAs combines the Büchi and the co-Büchi condition: existential sets refer to the Büchi condition G and universal sets refer to the co-Büchi condition B.

Given a transition function δ, let $\tilde{\delta}$ denote the dual function of δ. That is, for every q and σ with $\delta(q, \sigma) = \theta$, let $\tilde{\delta}(q, \sigma) = \tilde{\theta}$, where $\tilde{\theta}$ is obtained from θ by switching $\Box$ and $\Diamond$, switching $\vee$ and $\wedge$, and switching **true** and **false**. If, for example, $\theta = \Box p \vee (\textbf{true} \wedge \Diamond q)$ then $\tilde{\theta} = \Diamond p \wedge (\textbf{false} \vee \Box q)$,

Lemma 3. [MSS88,KVW00] *Given an HAA $\mathcal{A} = \langle \Sigma, Q, \delta, q_{in}, \langle G, B\rangle\rangle$, the alternating automaton $\tilde{\mathcal{A}} = \langle \Sigma, Q, \tilde{\delta}, q_{in}, \langle B, G\rangle\rangle$ is an HAA that complements $\mathcal{A}$.*

A *satellite* for an HAA $\mathcal{A} = \langle \Sigma, Q, \delta, q_{in}, \alpha\rangle$ is a deterministic word automaton $\mathcal{U} = \langle \Sigma, Q', \delta', q'_{in}\rangle$ with no acceptance condition. For a node $x = d_0 \cdots d_k$ of a Σ-labeled D-tree $\langle T, \tau\rangle$, let $word\_to(x) = \tau(\varepsilon) \cdot \tau(d_0) \cdot \tau(d_0 \cdot d_1) \cdot \tau(d_0 \cdot d_1 \cdots d_{k-1})$ be the word that labels the path from the root to x. When the HAA reads a node x of the input tree, its transitions may depend on $\delta'(q_{in}, word\_to(x))$, namely on the state in which $\mathcal{U}$ would have been if we had run it along the paths of the tree. Formally, the transition function of $\mathcal{A}$ is $\delta : Q \times \Sigma \times Q' \to \mathcal{B}^+(\Omega \times Q)$, and is such that when $\mathcal{A}$ is in state q as it reads the node x, it proceeds according to $\delta(q, \tau(x), \delta'(q'_{in}, word\_to(x)))$. Technically, an HAA with a satellite is equivalent to the HAA obtained by taking the product of $\mathcal{A}$ and $\mathcal{U}$: the new state space is $Q \times Q'$, and whenever the product is in state $\langle q, q'\rangle$ and reads a node x, the transition from $\langle q, q'\rangle$ is obtained from $\delta(q, \tau(x), q')$ by replacing each atom $\Box s$ or $\Diamond s$ by $\Box(s, \delta'(q', \tau(x)))$ or $\Diamond(s, \delta'(q', \tau(x)))$, respectively. The acceptance condition of the product HAA is induced by F. The partition of the state space to sets, the partial order on the sets, and their classification into transient, universal, and existential are induced by these in $\mathcal{A}$.

Note that satellites are only a convenient way to describe HAA in which the state space can be partitioned to two components, one of which is deterministic, independent of the other, has no influence on the acceptance, and runs on all the branches of the tree. In particular, Lemma 3 holds also for HAA with satellites. It is sometimes convenient to describe the HAA and its satellite separately. In addition to clarity, the separation to $\mathcal{A}$ and $\mathcal{U}$ enables a tighter analysis of the complexity of the nonemptiness problem. Recall that the solution of the emptiness problem for alternating automata involves alternation removal, which results in a nondeterministic automaton with exponentially many states. While the product of an HAA with n states and a satellite with n' states has nn' states, there is a need to pay the exponential price of alternation removal in the process of the nonemptiness check only for $\mathcal{A}$. Formally, we have the following.

Theorem 4. *The nonemptiness problem for an HAA with n states and a satellite with n' states can be solved in time $2^{O(n \log n' + n^2 \log n)}$.*

Proof: Let $\mathcal{A}$ be an HAA with n states, and let $\mathcal{U}$ be its satellite with n' states. We first claim that $\mathcal{A}$ is not empty iff it accepts a tree of branching degree n. The proof of the claim is similar to the linear branching degree property of μ-calculus [CE81,SE89]. Now, we can translate $\mathcal{A}$ to a nondeterministic parity tree automaton $\mathcal{A}'$ over trees of branching degree n, with $n^{O(n)}$ states, index $O(n)$ [MS95], and with the same satellite $\mathcal{U}$. By taking the product of $\mathcal{A}'$ and $\mathcal{U}$, we get a nondeterministic parity automaton with $n'n^{O(n)}$ states, index $O(n)$, and no satellite. Checking the nonemptiness of such an NPT requires time $n'^{O(n)} n^{O(n^2)} = 2^{O(n \log n' + n^2 \log n)}$ [EJ88,PR89]. □

Theorem 5. *The 1-letter nonemptiness problem for an HAA with n states, depth m, and a satellite with n' states can be solved in space $O(m \log^2(nn'))$.*

Proof: Let $\mathcal{A}$ be an HAA with n states and depth m, and let $\mathcal{U}$ be its satellite with n' states. We can translate $\mathcal{A}$ to an HAA with nn' states, depth m, and no satellite. By [KVW00], checking the 1-letter nonemptiness of such an HAA requires space $O(m \log^2(nn'))$. □

4.2 From mCTL* to HAA

In this section we describe a translation of mCTL* formulas to HAA. In the sequel, we use the translation in order to obtain satisfiability and model-checking decision procedures for mCTL*.

We first need some definitions. For an mCTL* formula ψ, let $sf(\psi)$ be the set of state subformulas of ψ. For two formulas θ and φ of ψ, we say that θ is *maximal* in φ if θ is a strict state subformula of φ and there exists no state formula "between them", namely, there exists no strict subformula ξ of φ such that θ is a strict subformula of ξ. We denote by $max(\varphi)$ the set of all formulas maximal in φ. For example, $max(A((X\neg p)U(EXq))) = \{\neg p, EXq\}$. Consider an mCTL* formula ψ and a computation tree $\langle T, \tau\rangle$. We say that a $2^{sf(\psi)}$-labeled tree $\langle T, g\rangle$ is *sound* for ψ if for all $x \in T$ and $\theta \in sf(\psi)$, we have that $x, \epsilon \models \theta$ iff $\theta \in g(x)$. Thus, a node x is labeled by exactly all the formulas in $sf(\psi)$ that are satisfied in x, with ϵ being the present (note that the fact ϵ is the present is important only for $\theta = present$).

Theorem 6. *Given an mCTL* formula ψ, we can construct an HAA $\mathcal{A}_\psi$ with $2^{O(|\psi|)}$ states, depth $O(|\psi|)$, and a satellite with $2^{2^{O(|\psi|)}}$ states, such that $\mathcal{A}_\psi$ runs on $2^{sf(\psi)}$-labeled trees and accepts exactly all trees that are sound for ψ and satisfy ψ.*

Proof: We first define the satellite $\mathcal{U}$ for $\mathcal{A}_\psi$. Consider a subformula of ψ of the form $E\xi$. Let $\mathcal{U}_\xi = \langle 2^{sf(\psi)}, Q_\xi, M_\xi, Q^{in}_\xi, \alpha_\xi\rangle$ be a nondeterministic Büchi automaton on infinite words such that $\mathcal{U}_\xi$ accepts exactly all the computations satisfying ξ [VW94]. Note that $\mathcal{U}_\xi$ regards the formulas maximal in φ as atomic propositions ($\mathcal{U}_\xi$ ignores the other formulas in $sf(\psi)$). Let $\mathcal{U}^d_\xi = \langle 2^{sf(\psi)}, 2^{Q_\xi}, M^d_\xi, \{Q^{in}_\xi\}\rangle$ be the deterministic automaton with no acceptance condition obtained by applying the subset construction [RS59] to $\mathcal{U}_\xi$. Thus, for all $S \in 2^{Q_\xi}$ and $\sigma \in 2^{sf(\psi)}$, we have that $M^d_\xi(S, \sigma) = \bigcup_{s\in S} M_\xi(s, \sigma)$. Now, the satellite $\mathcal{U} = \langle 2^{sf(\psi)}, Q', \delta', q'_{in}\rangle$ is the crossproduct of all the automata $\mathcal{U}^d_\xi$ above (for all the subformulas $E\xi$ of ψ). Intuitively, $\mathcal{U}$ supplies to $\mathcal{A}_\psi$ the information required in order to evaluate path formulas on paths that start in the root of the tree and visit the current node. When there is a need to check that $E\xi$ holds in some node x, the automaton $\mathcal{A}_\psi$ guesses a state q that is a member of the current state $S \in 2^{Q_\xi}$ of $\mathcal{U}^d_\xi$ (recall that 2^{Q_ξ} is a component in the state space of the satellite) and it executes $\mathcal{U}^q_\xi$ along some x-path. The position in which $\mathcal{A}_\psi$ starts the execution of $\mathcal{U}^q_\xi$ is the only position in which the atomic proposition $present$ holds. Note that a correct guess of q and a successful run of $\mathcal{U}^q_\xi$ along some x-path are possible iff there is a path that visits x and satisfies ξ, which corresponds to the semantics of $E\xi$.

As in the case of the HAA for CTL* [KVW00], we construct $\mathcal{A}_\psi$ by induction on the structure of ψ. With each subformula φ of ψ, we associate an HAA $\mathcal{A}'_\varphi$ composed from HAAs associated with formulas maximal in φ. We assume that the state sets of composed HAAs are disjoint (otherwise, we rename states). The HAA $\mathcal{A}'_\varphi$ assumes that the tree is sound for the formulas in $max(\varphi)$ and only checks the satisfaction of φ under this assumption. We then define $\mathcal{A}_\psi$ as the intersection of $\mathcal{A}'_\psi$ with an automaton that checks, by sending copies to the different $\mathcal{A}'_\varphi$ automata, that the input tree is indeed sound with respect to ψ.

- If $\varphi = p$ for $p \in AP$ or $p = present$, then $\mathcal{A}'_\varphi$ is the one-state HAA that goes to **true** when it reads σ with $p \in \sigma$ and goes to **false** otherwise.

– If $\varphi = \neg\varphi_1$, then $\mathcal{A}'_\varphi$ is $\tilde{\mathcal{A}}_{\varphi_1}$ — the HAA obtained by dualizing the HAA $\mathcal{A}_{\varphi_1}$ for φ_1. If $\varphi = \varphi_1 \vee \varphi_2$, then $\mathcal{A}'_\varphi = \langle 2^{sf(\psi)}, Q^1 \cup Q^2 \cup \{q_{in}\}, \delta, q_{in}, \langle G^1 \cup G^2, B^1 \cup B^2\rangle\rangle$, where $\mathcal{A}'_{\varphi_i} = \langle 2^{sf(\psi)}, Q^i, \delta^i, q^i_{in}, \langle G^i, B^i\rangle\rangle$, q_{in} is a new state and δ is defined as follows. For states in Q^1 and Q^2, the transition function δ agrees with δ^1 and δ^2, respectively. For the state q_{in} and for all $\sigma \in 2^{sf(\psi)}$ and $q' \in Q'$, we have $\delta(q_{in}, \sigma, q') = \delta(q^1_{in}, \sigma, q') \vee \delta(q^2_{in}, \sigma, q')$. Thus, in the state q_{in}, the HAA $\mathcal{A}'_\varphi$ sends all the copies sent by $\mathcal{A}'_{\varphi_1}$ or all the copies sent by $\mathcal{A}'_{\varphi_2}$. The singleton $\{q_{in}\}$ constitutes a transient set, with the ordering $\{q_{in}\} > Q_i$ for all the sets Q_i in Q^1 and Q^2.
– If $\varphi = E\xi$, where ξ is an mCTL* path formula, we proceed as follows. Let $\mathcal{U}_\xi = \langle 2^{sf(\psi)}, Q_\xi, M_\xi, Q^{in}_\xi, \alpha_\xi\rangle$. Recall that the state space Q' of the satellite $\mathcal{U}$ is the product of the state spaces of the deterministic automata for the path formulas. Thus, each state $q' \in Q'$ has a component for each of these automata, and in particular for $\mathcal{U}^d_\xi$. Consider a state $q' \in Q'$. Let $q'_{|\xi}$ be the state of $\mathcal{U}^d_\xi$ in q'. Note that $q'_{|\xi} \in 2^{Q_\xi}$. Then, $\mathcal{A}'_\varphi = \langle 2^{sf(\psi)}, Q_\xi, \delta', q_{in}, \langle \alpha_\xi, \emptyset\rangle\rangle$ is defined so that from its initial state q_{in}, it consults the satellite's state q' and executes $\mathcal{U}_\xi$ along a single path, starting from some state in q'_ξ. The proposition $present$ holds exactly at the node in which the execution of $\mathcal{U}_\xi$ starts, thus its first transition assumes that $present$ holds. Formally, for all $\sigma \in 2^{sf(\psi)}$ and $q' \in Q'$, we have

$$\delta'(q_{in}, \sigma, q') = \bigvee_{s \in q'_{|\xi}} \bigvee_{s' \in M_\xi(s, \sigma \cup \{present\})} \Diamond s'.$$

Also, for all $q \in Q_\xi$, we have $\delta'(q, \sigma, q') = \bigvee_{q_i \in M_\xi(q,\sigma)} \Diamond q_i$. If $M_\xi(q, \sigma) = \emptyset$, then $\delta(q, \sigma, q') =$ **false**. Note that the only transition in which the input from the satellite is taken into an account is the transition from q_{in}, where $\mathcal{A}'_\varphi$ chooses a state from $q_{|\xi}$ to proceed with. Note also that Q_ξ constitutes a single existential set. The HAA $\mathcal{A}'_\varphi$ accepts a $2^{sf()}$-labeled tree from node x iff x satisfies φ, assuming the input tree is sound for the formulas in $max(\varphi)$. The states in Q_ξ constitute an existential set of the HAA.

We now add to $\mathcal{A}'_\psi$ transitions that check that the input tree is indeed sound for ψ, thus the letter read at node x describes the set of formulas satisfied in x. For this purpose, we add a new state q_{check}. Whenever $\mathcal{A}_\psi$ is in state q_{check} and reads a letter σ, it sends copies sent from the initial states of the HAA $\mathcal{A}'_{\varphi_i} = \langle 2^{sf(\psi)}, Q^i, \delta^i, q^i_{in}, \langle G^i, B^i\rangle\rangle$, for all $\varphi_i \in \sigma$, sends copies sent from the initial states of the HAA $\tilde{\mathcal{A}}'_{\varphi_i} = \langle 2^{sf(\psi)}, \tilde{Q}^i, \tilde{\delta}^i, q^i_{in}, \langle B^i, G^i\rangle\rangle$, for all $\varphi_i \notin \sigma$, (and also sends a copy that stays in q_{check} to all the successors). Formally, for all $\sigma \in 2^{sf(\psi)}$ and $q' \in Q'$, we have

$$\delta(q_{check}, \sigma, q') = \Box q_{check} \wedge \bigwedge_{\varphi_i \in \sigma} \delta^i(q^i_{in}, \sigma, q') \wedge \bigwedge_{\varphi_i \notin \sigma} \tilde{\delta}^i(q^i_{in}, \sigma, q').$$

The HAA $\mathcal{A}_\psi = \langle 2^{sf(\psi)}, Q, \delta, q_{in}, \langle G, B\rangle\rangle$ is such that Q is the union of $\{q_{in}, q_{check}\}$ with the union of the state spaces of $\mathcal{A}'_\theta$, for $\theta \in sf(\psi)$. The initial state q_{in} checks for the satisfaction of ψ and for the soundness with respect to ψ, thus $\delta(q_{in}, \sigma, q') = \delta^\psi(q^\psi_{in}, \sigma, q') \wedge \delta(q_{check}, \sigma, q')$, where δ^ψ and q^ψ_{in} are the transition function and initial state of $\mathcal{A}'_\psi$. Finally, $G = \bigcup_{\varphi_i \in sf(\psi)} G^i$ and $B = \bigcup_{\varphi_i \in sf(\psi)} B^i$. The state q_{in} is transient and the state q_{check} constitute a singleton universal set of the HAA.

The arguments about the correctness of the construction, its size, and its depth, are similar to these in [KVW00], and are given in Appendix A.1 □

4.3 Satisfiability

Theorem 7. *The satisfiability problem for mCTL* is 2EXPTIME-complete.*

Proof: We start with the upper bound. Consider an mCTL*- formula ψ. By Theorem 6, there is an HAA $\mathcal{A}_\psi$ with $2^{O(|\psi|)}$ states, depth $O(|\psi|)$, and a satellite with $2^{2^{O(|\psi|)}}$ states such that $\mathcal{L}(\mathcal{A}_\psi)$ is exactly the set of $2^{sf(\psi)}$-labeled trees that are sound for ψ and satisfy ψ. The HAA $\mathcal{A}_\psi$ is nonempty iff ψ is satisfiable. By Theorem 4, the nonemptiness of $\mathcal{A}_\psi$ can be decided in time $2^{2^{O(|\psi|)}}$, which is therefore also the time required for deciding the satisfiability of ψ.

It is left to prove the lower bound. By Theorem 1, each CTL* formula can be linearly translated to an equivalent mCTL* formula. The lower bound then follows from the fact CTL* satisfiability is 2EXPTIME-hard [VS85]. □

Note that the application of the 2EXPTIME lower bound of CTL* satisfiability is not possible for mCTL*-, and the exact complexity of the satisfiability problem for this logic is left open.

4.4 Model Checking

For CTL*, the automata-theoretic approach is similar for satisfiability and model checking: the HAA for a CTL* formula ψ accepts exactly all 2^{AP}-labeled trees that satisfy ψ, so satisfiability is reduced to emptiness and model checking to 1-letter emptiness (that is, emptiness for an automaton with a singleton alphabet) of the product of the HAA with the system [KVW00]. In the case of mCTL*, the need to execute the word automata for the path formulas from the root of the tree, has forced us to define the HAA with respect to $2^{sf(\psi)}$-labeled trees. While this was not a problem for satisfiability, it is a problem for model checking, where we need to take the product of the HAA with a Kripke structure that is labeled by 2^{AP}. Fortunately, this is not a real problem, as it is possible to guess an extension of the 2^{AP}-labeling to a $2^{sf(\psi)}$-labeling, and then let the HAA check the soundness of the guess.

Theorem 8. *Given an mCTL* formula ψ, we can construct an HAA $\mathcal{A}^{AP}_\psi$ with $2^{O(|\psi|)}$ states, depth $O(|\psi|)$, and a satellite with $2^{2^{O(|\psi|)}}$ states, such that $\mathcal{A}^{AP}_\psi$ runs on 2^{AP}-labeled trees and accepts exactly all trees that satisfy ψ.*

Proof: The construction is similar to the one described in Theorem 6, only that we have to adjust the HAA and its satellite to the alphabet 2^{AP}. Intuitively, instead of having the input tree labeled by subsets of $sf(\psi)$, we let the satellite guess the richer labels, and then let the HAA check the guess. Thus, the satellite is nondeterministic — on top of its deterministic transition we add a guess of the subset of $sf(\psi)$ to be read in the successor node. Yet, running the HAA on a 2^{AP}-labeled tree, and letting it check the guesses, guarantees that $word\_to(x)$ can be viewed as a word in $(2^{sf(\psi)})^*$ rather than a word in $(2^{AP})^*$. Accordingly, $\delta'(q_{in}, word\_to(x))$ is a singleton, as in the case of a deterministic satellite.

Formally, if in the construction in Theorem 6 we ended up with a satellite $\mathcal{U} = \langle 2^{sf(\psi)}, Q', \delta', q'_{in}\rangle$ and HAA $\mathcal{A}_\psi = \langle 2^{sf(\psi)}, Q, \delta, q_{in}, \langle G, B\rangle\rangle$, now we have a satellite $\mathcal{U}_{AP} = \langle 2^{AP}, Q' \times 2^{sf(\psi)}, \delta'_{AP}, q'_{in}\rangle$ where for all $\langle q', \sigma\rangle \in Q' \times 2^{sf(\psi)}$ and $\sigma' \in 2^{AP}$, we have $\delta'_{AP}(\langle q', \sigma\rangle, \sigma') = \{\delta'(q', \sigma)\} \times 2^{sf(\psi)}$, and $\mathcal{A}^{AP}_\psi = \langle 2^{AP}, Q, \delta^{AP}, q_{in}, \langle G, B\rangle\rangle$, where for all $q \in Q$, $\sigma' \in 2^{AP}$, and $\langle q', \sigma\rangle \in Q' \times 2^{sf(\psi)}$, we have $\delta^{AP}(q, \sigma', \langle q', \sigma\rangle) = \delta(q, \sigma, q') \wedge \bigwedge_{\varphi_i \in \sigma} \delta^i(q^i_{in}, \sigma, q') \wedge \bigwedge_{\varphi_i \notin \sigma} \tilde{\delta}^i(q^i_{in}, \sigma, q')$. □

Theorem 9. *The model-checking problem for mCTL* is EXPSPACE-complete.*

Proof: We start with the upper bound. Consider an mCTL* formula ψ. By Theorem 6, there is an HAA $\mathcal{A}_\psi$ with $2^{O(|\psi|)}$ states, depth $O(|\psi|)$, and a satellite with $2^{2^{O(|\psi|)}}$ states, such that $\mathcal{L}(\mathcal{A}_\psi)$ is exactly the set of computation trees satisfying ψ. Consider a Kripke structure K. By [KVW00], K satisfies ψ iff the 1-letter HAA obtained by taking the product of K with $\mathcal{A}_\psi$ is not empty. The product has $|K|2^{O(|\psi|)}$ states, depth $O(|\psi|)$, and a satellite with $2^{2^{O(|\psi|)}}$ states. Thus, by Theorem 5, its 1-letter nonemptiness problem can be solved in space $|\psi| \log^2(|K|2^{2^{O(|\psi|)}}) = |\psi|(\log^2 |K| + 2^{O(|\psi|)} \log |K| + 2^{O(\psi)})$. Note that the complexity is only logarithmic in the size of the structure, and the exponential dependency is in the length of the formula, which is usually much smaller.

In order to prove the EXPSPACE lower bound, we do a reduction from the *exponential tiling* problem. In this problem, we are given a fixed set T of tiles, two relations $H, V \subseteq T \times T$, an integer n, a tuple of n tiles $\langle t_0, \ldots, t_{n-1}\rangle \in T^n$, and a tile $t_{fin} \in T$. The problem is to decide whether there is $m \geq 0$ such that it is possible to tile a $(2^n \times m)$-square so that horizontal neighbors belong to H, vertical neighbors belong to V, the first n tiles in the first row are $t_o, \ldots, t_{n-1}$, and the first tile in the last row is t_{fin}. Thus, formally, a legal tiling is a function $t : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, m - 1\} \to T$ such that: (1) for all $0 \leq i \leq 2^n - 2$ and $0 \leq j \leq m - 1$, we have that $H(t(i, j), t(i + 1, j))$, (2) for all $0 \leq i \leq 2^n - 1$ and $0 \leq j \leq m - 2$, we have that $V(t(i, j), t(i, j + 1))$, (3) for all $0 \leq j \leq n - 1$, we have that $t(0, j) = t_j$, and (4) $t(0, m - 1) = t_{fin}$. The exponential tiling problem is known to be EXPSPACE-complete [Lew78].

Given a tiling problem $\mathcal{T} = \langle T, H, V, n, t_0, \ldots, t_{n-1}, t_{fin}\rangle$, we construct an mCTL* formula φ over a fixed set AP of atomic propositions such that a legal tiling for $\mathcal{T}$ exists iff the universal model M_{AP} for AP (that is, a clique in which each node is labeled by a different subset of AP) satisfies φ. The formula φ is of length polynomial in $\mathcal{T}$, and the size of M_{AP} is fixed, so we actually prove that the EXPSPACE lower bound holds for structures of a fixed size.

Some of the atomic propositions in AP encode tiles in T. We refer to a word $\pi \in (2^{AP})^\omega$ as an attempt to encode a legal tiling t. We divide the word π to blocks of length n. Every block corresponds to a single location (in which we place a single tile) in the $(2^n \times m)$-square. An atomic proposition in AP that acts as a 2^n-counter encodes the column $i \in \{0, \ldots, 2^n - 1\}$ of the location. The tile in each location is encoded in the first point in the block. In addition, we mark the first point of each block by an atomic proposition b, and mark the first point of blocks that encode locations with $i = 2^n - 1$ by an atomic proposition l. Proper increase of the counter as well as correct labelling of b and l can be forced by an LTL formula ξ.

Let $t_0 \ldots t_{2^n-1}, t'_0 \ldots t'_{2^n-1}$ be two successive rows of the tiling t. For each $0 \leq i \leq 2^n - 1$ we know, given t_i, the possible values for t_{i+1} (these for which $H(t_i, t_{i+1})$, in case $i < 2^n - 1$) and the possible values for t'_i (these for which $V(t_i, t'_i)$). Consistency with

H and V gives us a necessary condition for a word to encode a legal tiling. In addition, the tiling should satisfy the edge conditions; it should start with t_{init} and has t_{fin} in some position in which the value of the counter is 0. It is easy to come up with an LTL formula θ that guarantees the satisfaction of the conditions on horizontal neighbors and the edge conditions.

The difficult part in the reduction is in guaranteeing that the condition for vertical neighbors, which are exponentially far, holds. This is where the memoryfullness of φ comes into the picture. By pointing to $t[i,j]$ in the present (using $present$), we can relate it with $t[i,j-1]$. Indeed, $t[i,j-1]$ is encoded in a point in which a block that has the same counter value as the current block starts, and for which there is exactly one point in which l holds before we reach the present.

The formula φ is then of the form

$$E[\xi \wedge \theta \wedge G(b \rightarrow EF(b \wedge \eta_0 \wedge \bigwedge_{1 \leq i \leq n} \eta_1(i) \wedge \eta_2)],$$

stating that the structure contains a path satisfying ξ and θ and in which whenever a block starts (this is the b on the left hand side of the $\rightarrow$), the path from the root to the present has a point in which a block starts (this is the b in the conjunct in the scope of EF), this block is in the previous row (this is enforced by $\eta_0 = ((\neg l)U(l \wedge X((\neg l)U present)$, which requires a single l between this b and the present), it has the same counter value as the block that starts now (this is enforced by η_1, which also uses $present$, and refers to the values of the counter bit by bit), and the tiles encoded in this block and the block in the present are allowed by V (this is enforced by η_2, which is a disjunction on the values allowed by V, with respect to the point in which b holds (encoding the value of the left element of the pair) and the point in which $present$ holds (encoding the value of the right element)).

If M_{AP} satisfies φ, then the path satisfying its existential requirement encodes a legal tiling for $\mathcal{T}$. Likewise, a legal tiling for $\mathcal{T}$ can be encoded in a path of M_{AP} that satisfies the existential requirement in φ. Accordingly, M_{AP} satisfies φ iff there is a legal tiling for $\mathcal{T}$. □

Of special interest is the mCTL* formula $AGE\xi$, for an LTL formula ξ. As discussed in Section 1, the formula is useful for checking possibility properties as well as in strong cyclic planning. In the full version, we show that our lower bound proof holds also for formulas of this form. The proof is based on the fact that pointing to the present with $present$ can be emulated by pointing to a location in the computation in which some special event occurs. In more detail, we add to the set of atomic propositions a special proposition $\#$, and instead of the universal model M_{AP}, we take two copies of M_{AP}, such that $\#$ does not hold in the states of the first copy and holds in all the states of the second copy. The formula we check is $EFA\xi$ (that is, the dual of possibility properties), where a path that satisfies the existential requirement encodes a legal tiling. Checking of vertical neighbors in all counters is done by referring to the value of the counter in the position in which the value of $\#$ flips from **false** to **true**. It follows that the EXPSPACE lower bound applies also for model checking of the less expressive mCTL*-, the Pistore-Vardi logic, and possibility properties.

We note that since CTL* model checking is PSPACE complete, the EXPSPACE lower bound for mCTL* model checking implies that mCTL*is exponentially more succinct

than CTL*. Also, since mCTL*- can be linearly translated to CTL$^\star_{lp}$, the latter suggests an alternative proof to the fact CTL$^\star_{lp}$ is exponentially more succinct than CTL* [Mar03]. In addition, it implies an EXPSPACE lower bound for the model-checking problem of CTL$^\star_{lp}$, a problem that was left open in [KP95] (see also [LS00]).

5 Discussion

We introduced and study mCTL* — a variant of CTL* in which path quantification ranges over paths that start in the beginning of the execution of the system and go through the present. We argued for the appropriateness of mCTL* for sanity checks and for planning in a nondeterministic domain, and showed that memoryfull path quantification can lead to formulas that are exponentially more succinct. Studying the algorithmic properties of memoryfull path quantification, we showed that while the transition from memoryless to memoryfull path quantification does not make the satisfiability problem harder, it does make the model-checking problem exponentially harder.

The fragment CTL of CTL* has received a lot of attention, and its model-checking problem can be solved in linear time [CES86]. One can also define the logic mCTL, which is the memoryfull counterpart of CTL. Thus, mCTL is the fragment of mCTL* in which every temporal operator is immediately preceded by a path quantifier. The logic mCTL- is then the fragment of mCTL in which the atomic proposition $present$ is not allowed. Unfortunately, mCTL and mCTL- are not of much interest. As we show in the full version, their expressive power is very limited (essentially, the fact its path formulas cannot contain a Boolean assertion containing $present$ makes mCTL as expressive as mCTL-, where present is lost with each application of a temporal operator), and still, their model-checking problem is at least NP-hard (the proof is similar to the NP-harness for the model checking of the fragment of LTL in which the only temporal operator is F [SC85]).

References

[BGK03] D. Berwanger, E. Grädel, and S. Kreutzer. Once upon a time in a west - determinacy, definability, and complexity of path games. In *Proc. 10th International Conference on Logic for Programming Artificial Intelligence and Reasoning*, volume 2850 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2003.

[BK98] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Ann. of Mathematics and Artificial Intelligence*, 22:5–27, 1998.

[BK00] F. Bacchus and F. Kabanza. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence*, 116(1–2):123–191, 2000.

[CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, January 1986.

[CM98] S. Cerrito and M.C. Mayer. Bounded model search in linear temporal logic and its application to planning. In *Proc. of 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 124–140. Springer Verlag, 1998.

[CRT98a] A. Cimatti, M. Roveri, and P. Traverso. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *Proc. of 15th National Conf. on Artificial Intelligence (AAAI'98)*, pages 875–881. AAAI Press, 1998.

[CRT98b] A. Cimatti, M. Roveri, and P. Traverso. Strong planning in non-deterministic domains via model checking. In *Proc. of 4th Int. Conf. on Artificial Intelligence Planning Systems (AIPS-98)*, pages 36–43. AAAI-Press, 1998.

[dGV99] G. de Giacomo and M.Y. Vardi. Automata-theoretic approach to planning with temporally extended goals. In *Proc. of 5th European Conf. in Planning (ECP'99)*, volume 1809 of *LNAI*, pages 226–238. Springer Verlag, 1999.

[DLPT02] U. Dal Lago, M. Pistore, and P. Traverso. Planning with a language for extended goals. In *Proc. of 18th National Conf. on Artificial Intelligence (AAAI'02)*. AAAI Press, 2002.

[DTV99] M. Daniele, P. Traverso, and M.Y. Vardi. Strong cyclic planning revisited. In S. Biundo and M. Fox, editors, *5th European Conference on Planning*, pages 34–46, 1999.

[EH86] E.A. Emerson and J.Y. Halpern. Sometimes and not never revisited: On branching versus linear time. *Journal of the ACM*, 33(1):151–178, 1986.

[EJ88] E.A. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 328–337, White Plains, October 1988.

[EL85] E.A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. In *Proc. 20th ACM Symp. on Principles of Programming Languages*, pages 84–96, New Orleans, January 1985.

[ES84] A.E. Emerson and A.P. Sistla. Deciding full branching time logics. *Information and Control*, 61(3):175–201, 1984.

[FLS02] N. Markey F. Laroussinie and Ph. Schnoebelen. Temporal logic with forgettable past. In *Proc. 17th IEEE Symp. on Logic in Computer Science*, pages 383–392, 2002.

[FN71] R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[HT87] T. Hafer and W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In *Proc. 14th International Coll. on Automata, Languages, and Programming*, volume 267 of *Lecture Notes in Computer Science*, pages 269–279. Springer, 1987.

[JW95] D. Janin and I. Walukiewicz. Automata for the modal μ-calculus and related results. In *Proc. 20th International Symp. on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 552–562. Springer, 1995.

[KD01] J. Kvarnström and P. Doherty. TALplanner: a temporal logic based forward chaining planner. *Ann. of Mathematics and Artificial Intelligence*, 30:119–169, 2001.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KP95] O. Kupferman and A. Pnueli. Once and for all. In *Proc. 10th IEEE Symp. on Logic in Computer Science*, pages 25–35, San Diego, June 1995.

[KV03] O. Kupferman and M.Y. Vardi. Vacuity detection in temporal model checking. *Journal on Software Tools For Technology Transfer*, 4(2):224–233, February 2003.

[KVW00] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, March 2000.

[Lam98] L. Lamport. Proving possibility properties. *Theoretical Computer Science*, 206(1–2):341–352, 1998.

[Lew78] H.R. Lewis. Complexity of solvable cases of the decision problem for the predicate calculus. In *Foundations of Computer Science*, volume 19, pages 35–47, 1978.

[LS00] F. Laroussinie and Ph. Schnoebelen. Specification in CTL+past for verification in CTL. *Information and Computation*, 156(1-2):236–263, January 2000.

[Mar03] N. Markey. Temporal logic with past is exponentially more succinct. *EATCS Bulletin*, 79:122–128, 2003.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 1980.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, Berlin, January 1992.

[MS87] D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[MS95] D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141:69–107, 1995.

[MSS88] D.E. Muller, A. Saoudi, and P. E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proceedings 3rd IEEE Symp. on Logic in Computer Science*, pages 422–427, Edinburgh, July 1988.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. on Foundation of Computer Science*, pages 46–57, 1977.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, Austin, January 1989.

[PS92] M. Peot and D. Smith. Conditional nonlinear planning. In *Proc. of 1st Int. Conf. on AI Planning Systems (AIPS'92)*, pages 189–197. Morgan Kaufmann Publisher, 1992.

[PT01] M. Pistore and P. Traverso. Planning as model checking for extended goals in non-deterministic domains. In *Proc. of 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01)*. AAAI Press, 2001.

[PV03] M. Pistore and M. Vardi. The planning spectrum – one, two, three, infinity. In *18th IEEE Symposium on Logic in Computer Science*, pages 234–243, Ottawa, Canada, June 2003. IEEE, IEEE press.

[PW92] J. Penberthy and D. Wed. UCPOP: A sound, complete, partial order planner for adl. In *Proc. of 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, 1992.

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

[SC85] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logic. *Journal ACM*, 32:733–749, 1985.

[SE89] R.S. Streett and E.A. Emerson. An automata theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, 81(3):249–264, 1989.

[Tho90] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 1990.

[Var01] M.Y. Vardi. Branching vs. linear time: Final showdown. In *Proc. 7th International Conference on Tools and algorithms for the construction and analysis of systems*, volume 2031 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2001.

[VS85] M.Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *Proc 17th ACM Symp. on Theory of Computing*, pages 240–251, 1985.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

[War76] D.H.D. Warren. Generating conditional plans and programs. In *Proc. of the Summer Conf. on Artificial Intelligence and Simulation of Behaviour (AISB'76)*, pages 344–354, 1976.

[Wil99] T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In C. Pandu Ragan, V. Raman, and R. Ramanujam, editors, *Proc. 19th conference on Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 1999.

[Wol81] P. Wolper. Temporal logic can be more expressive. In *Proc. 22nd IEEE Symp. on Foundations of Computer Science*, pages 340–348, Nashville, October 1981.

A Proofs

A.1 Correctness of the construction in Theorem 6

We prove the correctness of the construction by induction on the structure of φ. The proof is immediate for the case φ is of the form p, $\neg p$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, or $A\xi$. We consider here the case where $\varphi = E\xi$. If a node x in a tree $\langle T, \tau \rangle$ satisfies φ with c being the present, then there exists a path π in $\langle T, \tau \rangle$ such that $x \in \pi$ and $\pi \models \xi$ with c being the present. Thus, there exists an accepting run r of $\mathcal{U}_\xi$ on a word w that agrees with π on the formulas in $max(\varphi)$. Let s be the state of $\mathcal{U}_\xi$ that r visits when it reads the position of w that corresponds to node x. By the definition of $\mathcal{U}^d_\xi$, the state s is a member of the set S_ξ — the component of the state $q' \in Q$ that $\mathcal{U}$ visits when it reads node x. The HAA $\mathcal{A}_\psi$ visits nodes of the tree in the present. Hence, since r is accepting, there is a run of $\mathcal{U}_\xi$ that starts in state s on the suffix of w that starts in the position that corresponds to x, and in which $present$ holds when the node x is read. It follows that a run of $\mathcal{A}_\varphi$ that chooses to proceed with s and agrees with r on the x-path that corresponds to the suffix of π can accept $\langle T, \tau \rangle$ from node x. Indeed, by the definition of $\mathcal{A}'_\varphi$, the copies that proceeds according to δ' satisfy the acceptance condition. In addition, by the induction hypothesis, copies sent by q_{check} fulfill the acceptance condition. Now, if a run r of $\mathcal{A}_\varphi$ accepts a tree $\langle T, \tau \rangle$ from node x, then there must be a state s such that $\mathcal{U}_\xi$ can reach s when reads the word that labels the path from the root to x, and $\mathcal{U}_\xi$ accepts, from state s, the word that labels some x-path. Thus, the path π obtained by concatenating the path from the root to x and the above x-path is accepted by $\mathcal{U}_\xi$, and therefore, it satisfies ξ. Hence, by the semantics of mCTL*-, the node x of $\langle T, \tau \rangle$ satisfies φ.

We now consider the size of $\mathcal{U}$ and $\mathcal{A}_\psi$. We start with the size of the satellite $\mathcal{U}$. Since the size of $\mathcal{U}_\xi$ is exponential in $|\xi|$ [VW94] and the subset construction involves an additional exponential blow up, the size of $\mathcal{U}^d_\psi$ is doubly exponential in $|\xi|$. Hence, the size of $\mathcal{U}$ is at most doubly exponential in $|\psi|$. We now turn to the size of $\mathcal{A}_\psi$. For every φ, we prove, by induction on the structure of φ, that the size of $\mathcal{A}_\varphi$ is exponential in $|\varphi|$.

- Clearly, for $\varphi = p$ for some $p \in AP$, the size of $\mathcal{A}_\varphi$ is constant.
- For $\varphi = \neg\varphi_1$, we have $|\mathcal{A}_\varphi| = |\mathcal{A}_{\varphi_1}|$. By the induction hypothesis, $|\mathcal{A}_{\varphi_1}|$ is exponential in $|\varphi_1|$. Thus, $|\mathcal{A}_\varphi|$ is surely exponential in $|\varphi|$.
- For $\varphi = \neg\varphi_1$ or $\varphi = \varphi_1 \vee \varphi_2$, we have $|\mathcal{A}_\varphi| = O(|\mathcal{A}_{\varphi_1}| + |\mathcal{A}_{\varphi_2}|)$. By the induction hypothesis, $|\mathcal{A}_{\varphi_1}|$ is exponential in $|\varphi_1|$ and $|\mathcal{A}_{\varphi_2}|$ is exponential in $|\varphi_2|$. Thus, $|\mathcal{A}_\varphi|$ is surely exponential in $|\varphi|$.
- For $\varphi = E\xi$, we know, by [VW94], that the size of the word automaton $\mathcal{U}_\xi$ is exponential in $|\xi|$. Therefore, $\mathcal{A}'_\varphi$ is exponential in $|\varphi|$. Also, $|\Sigma'|$ is exponential in $|max(\varphi)|$ and, by the induction hypothesis, for all $\varphi_i \in max(\varphi)$, the size of $\mathcal{A}_{\varphi_i}$ is exponential in $|\varphi_i|$. Therefore, $\mathcal{A}_\varphi$ is also exponential in $|\varphi|$.

Finally, since each subformula of ψ induces exactly one set, the depth of $\mathcal{A}_\psi$ is linear in $|\psi|$.