Augmenting Branching Temporal Logics with Existential Quantification over Atomic Propositions*

Orna Kupferman
EECS Department
UC Berkeley
Berkeley CA 94720-1770, U.S.A.
Email: orna@eecs.berkeley.edu

June 4, 1997

Abstract

In temporal-logic model checking, we verify the correctness of a program with respect to a desired behavior by checking whether a structure that models the program satisfies a temporal logic formula that specifies this behavior. One of the ways to overcome the expressiveness limitation of temporal logics is to augment them with quantification over atomic propositions. In this paper we consider the extension of *branching temporal logics* with *existential quantification* over atomic propositions. Once we add existential quantification to a branching temporal logic, it becomes sensitive to unwinding. That is, unwinding a structure into an infinite tree does not preserve the set of formulas it satisfies. Accordingly, we distinguish between two semantics, two practices as specification languages, and two versions of the model-checking problem for such a logic. One semantics refers to the structure that models the program, and the second semantics refers to the infinite computation tree that the program induces. We examine the complexity of the model-checking problem in the two semantics for the logics CTL and CTL* augmented with existential quantification over atomic propositions. Following the cheerless results that we get, we examine also the program complexity of model checking; i.e., the complexity of this problem in terms of the program, assuming the formula is fixed. We show that while fixing the formula dramatically reduces model-checking complexity in the tree semantics, its influence on the structure semantics is poor.

*This research was supported in part by the Office of Naval Research Young Investigator award N00014-95-1-0520, by the National Science Foundation CAREER award CCR-9501708, by the National Science Foundation grant CCR-9504469, by the Air Force Office of Scientific Research contract F49620-93-1-0056, by the Army Research Office MURI grant DAAH-04-96-1-0341, by the Advanced Research Projects Agency grant NAG2-892, and by the Semiconductor Research Corporation contract 95-DC-324.036.

1 Introduction

Temporal logics, which are modal logics that enable the description of occurrence of events in time, serve as a classical tool for specifying behaviors of concurrent programs [21]. The appropriateness of temporal logics follows from the fact that finite-state programs can be modeled by finite *propositional Kripke structures*, whose properties can be specified using propositional temporal logic. This yields fully-algorithmic methods for synthesis and for reasoning about the correctness of programs. A powerful such method is *model checking*. In model checking, we verify the correctness of a program with respect to a desired behavior by checking whether the program, modeled as a Kripke structure, satisfies (is a model of) the temporal logic formula that specifies this behavior. Recent methods and heuristics such as BDDs [3, 4], modular model checking [7, 13], partial-order techniques, [29], on the fly model checking [8, 2], and others, cope successfully with the known "state explosion" problem and give rise to model checking not only as a lovely theoretical issue, but also as a practical tool used for formal verification.

Model-checking methods consider two types of temporal logics: linear and branching [17]. In *linear temporal logics*, each moment in time has a unique possible future. Accordingly, linear temporal logic formulas are interpreted over a path in a Kripke structure and refer to a single computation of a program. In *branching temporal logics*, each moment in time may split into several possible futures. Accordingly, branching temporal logic formulas are interpreted over a state in a Kripke structure and refer to all the computations that start at this state. The syntax of the logic controls the way in which these computations can be referred to and determines the *expressive power* of the logic. Naturally, there is a trade-off between the expressive power of the logic and the *complexity* of its model-checking problem: the more a logic is expressive, the more expensive its model checking is.

Adding *quantification over atomic propositions* increases the expressive power of temporal logics [24, 25, 22]. In this paper, we consider the extension of branching temporal logics with *existential* quantification. Formally, if ψ is a formula in some branching temporal logic $\mathcal{L}$, then $\exists p_1 \ldots p_n \psi$, where $p_1, \ldots, p_n$ are atomic propositions, is a formula in the logic EQ$\mathcal{L}$, which augments $\mathcal{L}$ with existential quantification. The formula $\exists p_1 \ldots p_n \psi$ is satisfied in a Kripke structure K iff there exists a Kripke structure that satisfies ψ and differs from K in at most the labeling of $p_1, \ldots, p_n$.

The model-checking problem for EQ$\mathcal{L}$ stands somewhere between the model-checking and the satisfiability problems for $\mathcal{L}$. On the one hand, as in model checking, we are given both a Kripke structure and a formula and we are asked whether the structure satisfies the formula. On the other hand, as in satisfiability, we are asked about the existence of some Kripke structure that satisfies the formula. Essentially, we can view the model-checking problem for EQ$\mathcal{L}$ as a restricted (or perhaps extended) version of the satisfiability problem for $\mathcal{L}$, in which the candidates to satisfy the formula are not all Kripke structures, but only a limited subset of them. Here, naturally enough, comes the question of complexity. The satisfiability problem for a branching temporal logic $\mathcal{L}$ is usually harder than its model-checking problem. For example, the branching temporal logics CTL and CTL* have, respectively, EXPTIME and 2EXPTIME complete satisfiability bounds [12, 26, 11, 9] and have, respectively, linear-time and PSPACE-complete bounds for their model checking problems [6, 10]. Where does the complexity of the model-checking problem for EQ$\mathcal{L}$ stand? Is it necessarily between the complexities of the model-checking problem and the satisfiability problem for $\mathcal{L}$? To which of them is it closer? Is it worth paying the increase in model-checking complexity for the increase in the expressive power?

A key observation that should be made before answering these questions is that once we add existential quantification to a branching temporal logic $\mathcal{L}$, it becomes *sensitive to unwinding*. That is, unwinding of a Kripke structure into an infinite computation tree does not preserve the set of EQ$\mathcal{L}$ formulas it satisfies. Consequently, we distinguish between two semantics for EQ$\mathcal{L}$. The first is the structure semantics given above. The second, which we call EQ$\mathcal{L}_t$, corresponds to a tree semantics. According to this semantics, a Kripke structure K satisfies a formula $\exists p_1 \ldots p_n \psi$ iff there exists a computation tree that satisfies ψ and differs from the computation tree obtained by unwinding K in at most the labels of $p_1, \ldots, p_n$. Intuitively, it is harder for K to satisfy a formula in the structure semantics: among the infinitely many computation trees that we have as candidates for satisfaction in the tree semantics, only finitely many, these in which nodes that correspond to the same state of K have the same labeling, are candidates in the structure semantics. The logics EQ$\mathcal{L}$ and EQ$\mathcal{L}_t$ differ in their practices as specification languages, differ in their expressive power, and differ in their model-checking complexities. Nevertheless, we found in the literature unawareness to this sensitivity.

We show that existential quantification increases the expressive power of CTL and CTL*, in both semantics. In particular, existential quantification in the tree semantics is strong enough to replace *satellites*. A satellite, as introduced in [1], is a small finite state machine, linked to a design to be verified. It can read the design at any moment and it records particular events of interest, for possible use in the specification of the design. A concept similar to satellites is introduced in [19] as *observer processes*. For example, we can define a satellite $Raise(s)$ which detects cycles in which the signal s is raised. Satellites overcome the expressiveness limitations of CTL and are used successfully as a part of the formal-verification system in IBM Haifa. The price of satellites is the increase in the state space, which now consists of the product of the state space of the design with the state space of the satellite. Existential quantification leaves the design clean and shifts this price to the specification. For example, instead of checking a CTL formula ψ which requires the activation of the satellite $Raise(s)$, we can check the EQCTL$_t$ formula obtained by taking the conjunction of ψ with $\exists q AG(s \rightarrow AXq) \wedge AG(\neg s \rightarrow AX \neg q)$ and replacing each occurrence of $Raise(s)$ by $s \wedge \neg q$. Note that the quantified proposition q labels a node iff s holds in its predecessor node. In fact, by [16], existential quantification is sufficient to express any occurrence of events in the past that can be expressed by linear temporal logic. In addition, we can use existential quantification to count y modulo z. The way we use formulas in the structure semantics is different. There, formulas describe a single computation which is a partially ordered set [20]. For example, the formula $\exists q(q \wedge AG(q \rightarrow AXAXq) \wedge AG(q \rightarrow send_i)$ specifies that process i sends a message in all its even positions.

We analyze the complexity of the model-checking problem for the logics EQCTL, EQCTL$_t$, EQCTL*, and EQCTL$^\star_t$. Lichtenstein and Pnueli argued that when analyzing the complexity of model checking, a distinction should be made between complexity in the size of the input structure and complexity in the size of the input formula; it is the complexity in size of the structure that is typically the computational bottleneck [18]. Following this approach, we consider also the *program complexity* [27] of model checking for these logics; i.e., the complexity of this problem in terms of the size of the input Kripke structure, assuming the formula is fixed. Our main results are summarized in the table below.

		No Quantification	Quantification with structure semantics	Quantification with tree semantics
CTL	model checking	linear time [6]	NP-complete [Theorem 4.1]	EXPTIME-complete [Theorem 4.2]
	program complexity	NLOGSPACE-complete [2]	NP-complete [HK94, Theorem 5.1]	P-complete [Theorem 5.2]
CTL*	model checking	PSPACE-complete [10]	PSPACE-complete [Theorem 4.1]	2EXPTIME-complete [Theorem 4.2]
	program complexity	NLOGSPACE-complete [2]	NP-complete [Theorem 5.1]	P-complete [Theorem 5.2]

Examining our results, we conclude the following. First, in the structure semantics, existential quantification takes the model-checking problem for CTL from P to NP-complete. Thus, we can not expect an algorithm that does better than a naive check of all the possible labelings for the quantified propositions. The same penalty (moving from a deterministic complexity class to its nondeterministic variant) applies also for CTL*. There, however, as PSPACE = NPSPACE, it seems we do not really pay for it. Second, in the tree semantics, existential quantification makes model-checking as hard as satisfiability (this holds for every branching temporal logic that satisfies the *small branching degree property*). We show that these results hold also for very limited fragments of EQCTL and EQCTL*; e.g., when the propositional assertions are in 2CNF or when only a single quantified proposition is allowed. In addition, we show that there are branching temporal logics $\mathcal{L}$ for which the model-checking problem for EQ$\mathcal{L}$ is harder than the satisfiability problem for $\mathcal{L}$. As for satisfiability, we show that for logics $\mathcal{L}$ that satisfy the *finite model property*, the satisfiability problems for EQ$\mathcal{L}$ and EQ$\mathcal{L}_t$ are as hard as the satisfiability problem for $\mathcal{L}$. Thus, as far as satisfiability is concerned, we can have existential quantification for free.

Things become surprising when we turn to consider the program complexity. Mysteriously, while model checking in the tree semantics is harder than model checking in the structure semantics, we have that the program complexity of model checking is lower in the tree semantics. The elucidation of this mystery lies in the fact that the model-checking problem for EQ$\mathcal{L}_t$ is closer to the satisfiability problem for $\mathcal{L}$ than the model-checking problem for EQ$\mathcal{L}$ is. While this disfavors the tree semantics when we consider model-checking complexity, it advantages the tree semantics when we fix the formula. It follows from our results that in the structure semantics, fixing the formula still leaves us with the naive algorithm that checks all possible labeling for the quantified propositions. In the tree semantics, we can apply automata-theoretic methods to obtain model-checking procedures which are polynomial in the size of the Kripke structure. We can not, however, reach the space-efficient program complexity of model checking for CTL and CTL*.

2 Preliminaries

The logic *CTL** combines both branching-time and linear-time operators. 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"). There are two types of formulas in CTL*: *state formulas*, whose satisfaction is related to a specific state, and *path formulas*, whose satisfaction is related to a specific path. Formally, let AP be a set of atomic proposition names. A CTL* state formula is either:

- **true**, **false**, or p, for all $p \in AP$.
- $\neg\varphi_1$ or $\varphi_1 \vee \varphi_2$, where φ_1 and φ_2 are CTL* state formulas.
- $E\psi_1$, where ψ_1 is a CTL* path formula.

A CTL* path formula is either:

- A CTL* 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 CTL* path formulas.

The logic CTL* consists of the set of state formulas generated by the above rules. We use the usual abbreviations $\wedge$ ("and"), $\rightarrow$ ("implies"), A ("for all paths"), F ("eventually"), and G ("always").

The logic *CTL* is a restricted subset of CTL* in which the temporal operators must be immediately preceded by a path quantifier. Formally, it is the subset of CTL* obtained by restricting the path formulas to be $X\varphi_1$, $\varphi_1 U \varphi_2$, or their negations, where φ_1 and φ_2 are CTL state formulas.

The semantics of CTL* is defined with respect to a *Kripke structure* $K = \langle AP, W, R, w^0, L\rangle$, 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 $R(w, w')$), w^0 is an initial state, and $L : W \rightarrow 2^{AP}$ maps each state to a set of atomic propositions true in this state. A *path* of K is an infinite sequence $\pi = w_0, w_1, w_2, \ldots$ of states such that for all $i \geq 0$ we have $R(w, w')$. For a path π and an index $j \geq 0$, we use π^j to denote the suffix $w_j, w_{j+1}, w_{j+2}, \ldots$ of π.

The notation $K, w \models \varphi$ indicates that a CTL* state formula φ holds at the state w of the Kripke structure K. Similarly, $K, \pi \models \psi$ indicates that a CTL* path formula ψ holds at a path π of the Kripke structure K. When K is clear from the context, we write $w \models \varphi$ and $\pi \models \psi$. Also, $K \models \varphi$ if and only if $K, w^0 \models \varphi$.

The relation $\models$ is inductively defined as follows.

- For all w, we have $w \models \mathbf{true}$ and $w \not\models \mathbf{false}$.
- $w \models p$ for $p \in AP$ iff $p \in L(w)$.
- $w \models \neg p$ for $p \in AP$ iff $p \notin L(w)$.
- $w \models \neg\varphi_1$ iff $w \not\models \varphi_1$
- $w \models \varphi_1 \vee \varphi_2$ iff $w \models \varphi_1$ or $w \models \varphi_2$.
- $w \models E\psi$ iff there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, such that $\pi \models \psi$.
- $\pi \models \varphi$ for a state formula φ, iff $w_0 \models \varphi$ where $\pi = w_0, w_1, \ldots$
- $\pi \models \neg\psi_1$ iff $\pi \not\models \psi_1$.
- $\pi \models \psi_1 \vee \psi_2$ iff $\pi \models \psi_1$ or $\pi \models \psi_2$.
- $\pi \models X\psi$ iff $\pi^1 \models \psi$.
- $\pi \models \psi_1 U \psi_2$ iff there exists $i \geq 0$ such that $\pi^i \models \psi_2$ and for all $0 \leq j < i$, we have $\pi^j \models \psi_1$.

Given two Kripke structures $K = \langle AP, W, R, w^0, L\rangle$ and $K' = \langle AP', W', R, w'^0, L'\rangle$, we say that K' is $\{p_1, \ldots, p_n\}$*-different* from K iff $AP' = AP \cup \{p_1, \ldots, p_n\}$, $W' = W$, $R' = R$, $w'^0 = w^0$, and for all $w \in W$ and $p \in AP \setminus \{p_1, \ldots, p_n\}$, we have that $p \in L'(w)$ iff $p \in L(w)$.

The logic *EQCTL** is obtained by adding existential quantification to CTL*: if ψ is a CTL* formula and $p_1 \ldots p_n$ are atomic propositions, then $\exists p_1 \ldots p_n \psi$ is an EQCTL* formula. The semantics of $\exists p_1 \ldots p_n \psi$ is given by $K \models \exists p_1 \ldots p_n \psi$ iff there exists a Kripke structure K', such that $K' \models \psi$ and K' is $\{p_1, \ldots, p_n\}$-different from K. Note that EQCTL* is not closed under negation. Thus, formulas of the form $\forall p_1 \ldots p_n \psi$ are not EQCTL* formulas. The logic *EQCTL* is defined similarly, by adding existential quantification to CTL.

Given a formula $\exists p_1 \ldots p_n \psi$, we call the atomic propositions $p_1 \ldots p_n$ *quantified propositions* and we call all the other propositions in ψ *free propositions*. Note that satisfaction of an EQCTL* formula with no free propositions in a Kripke structure K is independent of AP and L. A *frame* is a Kripke structure with no AP and L. A frame $K = \langle W, R, w^0\rangle$ satisfies an EQCTL* formula $\exists p_1 \ldots p_n \psi$ iff there exists a Kripke structure $K' = \langle AP, W, R, w^0, L\rangle$ such that $K' \models \psi$.

A *tree* is a set $T \subseteq \mathbb{N}^*$ such that if $x \cdot c \in T$ where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$, and for all $0 \leq c' < c$, we have that $x \cdot c' \in T$. The elements of T are called *nodes*, and the empty word ϵ is the *root* of T. Given an alphabet Σ, a *Σ-labeled tree* is a pair $\langle T, V\rangle$ where T is a tree and $V : T \to \Sigma$ maps each node of T to a letter in Σ. A *computation tree* is a Σ-labeled tree with $\Sigma = 2^{AP}$ for some set AP of atomic propositions.

3 Expressive Power

A Kripke structure K can be unwound into an infinite computation tree in a straightforward way. We denote by $\langle T_K, V_K\rangle$ the computation tree obtained from unwinding K. Formally, for every node w, let $d(w)$ denote the degree of w (i.e., the number of successors that w has, and note that for all w we have $d(w) \geq 1$), and let $succ_R(w) = \langle w_0, \ldots, w_{d(w)-1}\rangle$ be an ordered list of w's R-successors (we assume that the nodes of W are ordered). We first define the W-labeled tree $\langle T_K, V_K^w\rangle$ that corresponds to K inductively as follows:

1. $\varepsilon \in T_K$ and $V_K^w(\varepsilon) = w^0$.
2. For $y \in T_K$ with $succ_R(V_K^w(y)) = \langle w_0, \ldots, w_m\rangle$ and for $0 \leq i \leq m$, we have $y \cdot i \in T_K$ and $V_K^w(y \cdot i) = w_i$.

. Now, $\langle T_K, V_K\rangle$ is the computation tree obtained from $\langle T_K, V_K^w\rangle$ by taking the label of a node $x \in T_K$ to be $L(V_K^w(x))$ instead $V^w(x)$.

Each state in K may correspond to several nodes in $\langle T_K, V_K\rangle$. Since all these nodes have the same future (i.e., they are roots of identical subtrees) and since CTL can refer only to the future, CTL is *insensitive to unwinding.* That is, for every CTL formula φ and for every Kripke structure K, we have that $K \models \varphi$ iff $\langle T_K, V_K\rangle \models \varphi$. Insensitivity to unwinding is an important property for a branching temporal logic. Logics which are insensitive to unwinding, we can model check their formulas with respect to a finite Kripke structure, and adopt the result for its infinite computation tree. Symmetrically, we can model check an infinite computation tree using, say, automata-theoretic methods, and adopt the result for all Kripke structures that can be unwound into this tree. Augmenting CTL with past-time modalities, it becomes sensitive to unwinding. Since past-time modalities can be expressed by existential quantification [16], we have the following:

Theorem 3.1 *EQCTL is sensitive to unwinding.*

Proof: Consider the EQCTL formula $\varphi = \exists q AG(p \leftrightarrow AXq)$ and consider the Kripke structure

$$K = \langle \{p\}, \{w_0, w_1\}, \{\langle w_0, w_1\rangle, \langle w_1, w_1\rangle\}, w_0, \{\langle w_0, \{p\}\rangle, \langle w_1, \emptyset\rangle\}\rangle.$$

Since $p \in L(w_0)$ and since w_1 is a successor of w_0, it must be that q holds in the state w_1 of a Kripke structure that satisfies $AG(p \leftrightarrow AXq)$ and is $\{q\}$-different from K. On the other hand, since $p \notin L(w_1)$ and and since w_1 is the only successor of itself, it must be that q does not hold in the state w_1 of a Kripke structure that satisfies $AG(p \leftrightarrow AXq)$ and is $\{q\}$-different from K.

Thus, there exists no Kripke structure that satisfies $AG(p \leftrightarrow AXq)$ and is $\{q\}$-different from K. Hence, $K \not\models \varphi$. We now show that $\langle T_K, V_K \rangle \models \varphi$. Consider the computation tree $\langle T_K, V'_K \rangle$ over the alphabet $2^{\{p,q\}}$, where $V'_K(0) = \{p\}$, $V'_K(1) = \{q\}$, and for all $x \geq 2$, we have that $V'_K(x) = \emptyset$. Clearly, $\langle T_K, V'_K \rangle \models AG(p \leftrightarrow AXq)$ and thus, $\langle T_K, V_K \rangle \models \varphi$. $\square$

So, it makes sense to define two different semantics for EQCTL. The first corresponds to the original structure semantics and the second, which we call EQCTL$_t$, corresponds to a tree semantics. Precisely, an EQCTL$_t$ formula $\varphi = \exists p_1 \ldots p_n \psi$ is satisfied in a Kripke structure K, denoted $K \models_t \varphi$, iff there exists a computation tree $\langle T_K, V'_K \rangle$ such that $\langle T_K, V'_K \rangle \models \psi$ and V'_K differs from V_K in at most the labeling of $p_1, \ldots, p_n$; i.e., for every $x \in T_K$ and for every $p \in AP \setminus \{p_1, \ldots, p_n\}$, we have $p \in V_K(x)$ iff $p \in V'_K(x)$. Note that $K \models \varphi$ implies that $K \models_t \varphi$. It is the other direction which makes EQCTL sensitive to unwinding.

An interesting example for the sensitivity of EQCTL to unwinding is the formula $\varphi_1 = \exists q(q \wedge (AX\neg q) \wedge AG(q \leftrightarrow AXAXq) \wedge AG(q \rightarrow p))$. The formula is suggested in the literature for specifying the property $G2(p) =$ "p holds in all even places". When interpreted over computation trees, φ_1 indeed specifies $G2(p)$. To see this, note that the quantified proposition q holds in exactly all the even places. Yet, for a Kripke structure with a state that can be reached from the initial state by both an even number and an odd number of transitions (e.g., a Kripke structure that consists of a single state with a self loop), any labeling of q fails, even if this Kripke structure does satisfy $G2(p)$. Hence, φ_1 is appropriate only for the tree semantics.

We have just seen that EQCTL$_t$ is strong enough to specify $G2(p)$. In fact, the formula $\varphi_2 = \exists q(q \wedge AG(q \rightarrow AXAXq) \wedge AG(q \rightarrow p))$ specifies $G2(p)$ faithfully with respect to both the tree and the structure semantics. As opposed to φ_1, the formula φ_2 enables states which can be reached from the initial state by both an even and an odd number of transitions can be labeled with q. As CTL can not specify $G2(p)$ [28], we have the following:

Theorem 3.2 *EQCTL and EQCTL$_t$ are both strictly more expressive than CTL.*

Theorems 3.1 and 3.2 clearly hold also with respect to EQCTL*.

Insensitivity to the sensitivity of EQCTL and EQCTL* to unwinding hides also when comparing these logics with tree automata [11]. Indeed, EQCTL$^\star_t$ is as expressive as symmetric pair automata on infinite binary trees. Nevertheless, the translation of EQCTL$^\star_t$ into $2S2$, which is the base of this equivalence, does not hold for EQCTL*. Similarly, it is EQCTL$_t$, only, which is as expressive as symmetric Büchi automata on infinite binary trees.

4 Model-Checking Complexity

The model-checking problem for a variety of branching temporal logics can be stated as follows: given a branching temporal logic formula φ and a finite Kripke structure $K = \langle AP, W, R, w^0, L \rangle$, determine whether K satisfies φ. When some of the logics are sensitive to unwinding, there are two possible interpretations of this problem. The first interpretation, which is the one appropriate for EQCTL and EQCTL*, asks whether $K \models \varphi$. In the second interpretation, which is the one appropriate for EQCTL$_t$ and EQCTL$_t^\star$, we are given φ and K and are asked to determine whether $K \models_t \varphi$. In this section we consider model-checking complexity for the two interpretations.

Theorem 4.1

(1) *The model-checking problem for EQCTL is NP-complete.*

(2) *The model-checking problem for EQCTL* is PSPACE-complete.*

Proof: (1) We first prove membership in NP. In order to check whether a Kripke structure K satisfies an EQCTL formula $\exists p_1 \ldots p_n \psi$, we guess a Kripke structure K' that differs from K in at most the labeling of $p_1 \ldots p_n$, and then check, in linear time [6], whether K' satisfies the CTL formula ψ. To prove hardness in NP, we do a reduction from SAT. Clearly, a propositional formula ξ over the propositions $p_1 \ldots p_n$ is satisfiable if and only if the EQCTL formula $\exists p_1 \ldots p_n \xi$ is satisfied in a one-state frame.

(2) Both membership and hardness in PSPACE follow from being CTL* model checking PSPACE-complete [10]. While hardness is immediate, Savitch's Theorem [23] is required for the membership. $\square$

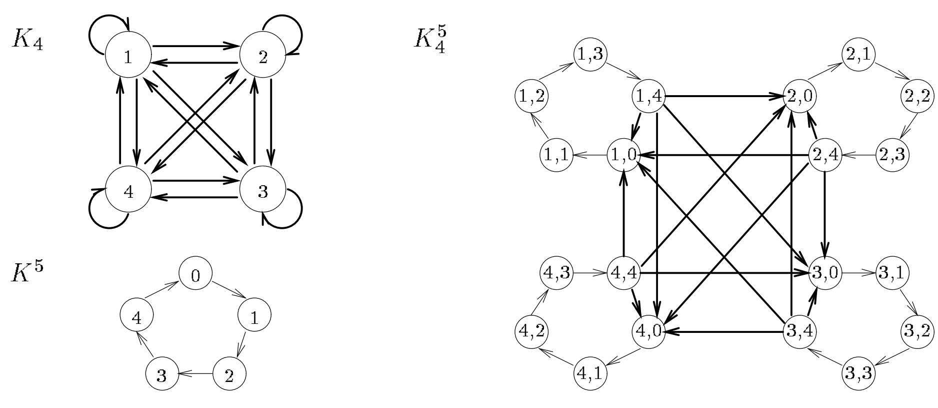

Figure 1: The frames K_4, K^5, and K_4^5.

Theorem 4.2

(1) *The model-checking problem for EQCTL$_t$ is EXPTIME-complete.*

(2) *The model-checking problem for EQCTL$_t^\star$ is 2EXPTIME-complete.*

Proof: (1) We first prove membership in EXPTIME. Given a set $\mathcal{D} \subset \mathbb{N}$ and an EQCTL$_t$ formula $\varphi = \exists p_1 \ldots p_n \psi$, let $\mathcal{A}_{\mathcal{D},\psi}$ be a Büchi tree automaton that accepts exactly all the tree models of ψ with branching degrees in $\mathcal{D}$. By [27], such $\mathcal{A}_{\mathcal{D},\psi}$ of size $O(|\mathcal{D}| * 2^{|\psi|})$ exists. Given a Kripke structure $K = \langle AP, W, R, w^0, L\rangle$ and a set S of atomic propositions, let $\mathcal{A}_{K,S}$ be a Buchi tree automaton that accepts exactly all the $(2^{AP \cup S})$-labeled trees $\langle T_K, V'_K\rangle$ for which V'_K differs from V_K in at most the labels of the propositions in S. It is easy to see that such $\mathcal{A}_{K,S}$ of size $O(|K| * 2^{|S|})$ exists. Taking $\mathcal{D}$ as the set of branching degrees in T_K and taking $S = \{p_1 \ldots p_n\}$, we get that $K \models_t \varphi$ iff $\mathcal{L}(\mathcal{A}_{K,S}) \cap \mathcal{L}(\mathcal{A}_{\mathcal{D},\psi}) \neq \emptyset$. By [27], the later can be checked in time $poly(|K| * 2^{|\varphi|})$.

For proving hardness in EXPTIME, we reduce the satisfiability problem for CTL, proved to be EXPTIME-hard in [12], to EQCTL$_t$ model checking. For every $m \geq 1$, let K_m denote the frame $\langle\{1, \ldots, m\}, \{1, \ldots, m\} \times \{1, \ldots, m\}, 1\rangle$. The frame K_4 is presented in Figure 1. Since a CTL formula ψ is satisfiable iff it is satisfied in a tree of branching degree $|\psi|$, and since unwinding $K_{|\psi|}$ results in such a tree, satisfiability of ψ can be reduced to model checking of $K_{|\psi|}$ with respect to the EQCTL$_t$ formula $\exists p_1 \ldots p_n \psi$, where $p_1 \ldots p_n$ are exactly all the atomic propositions in ψ.

(2) The model-checking procedure for EQCTL$_t^\star$ is similar to the one for EQCTL$_t$. Here, following [11], we have that $\mathcal{A}_{\mathcal{D},\psi}$ is a Rabin tree automaton with $2^{2^{|\psi|}}$ states and $2^{|\psi|}$ pairs. By [9], checking the nonemptiness of $\mathcal{L}(\mathcal{A}_{K,S}) \cap \mathcal{L}(\mathcal{A}_{\mathcal{D},\psi})$ can then be done in time $poly(|K| * 2^{2^{|\varphi|}})$. To prove hardness of EQCTL$_t^\star$ model checking in 2EXPTIME, we reduce satisfiability of CTL*, proved to be 2EXPTIME-hard in [26], to EQCTL$_t^\star$ model checking. Since a CTL* formula ψ is satisfiable iff it is satisfied in a tree of branching degree $|\psi|$, the same reduction that works for EQCTL$_t$ works also here. □

As CTL subsumes propositional logic, EQCTL model checking being NP-hard is far from being surprising. What, however, if we restrict CTL to subsume only a subset of propositional logic for which satisfiability is in P? Let 2CNF-EQCTL denote the subset of EQCTL in which the propositional assertions are in 2CNF.

Theorem 4.3 *The model checking problem for 2CNF-EQCTL is NP-hard.*

Proof: For every $n \geq 1$, let $\psi(n) = \bigwedge_{j \neq i} AG((\neg p_i) \vee (\neg p_j))$ where i and j range over $1 \ldots n$. For every Kripke structure K, we have that $K \models \psi(n)$ iff at most one p_i holds in each state of K. Note that all the propositional assertions in $\psi(n)$ are in 2CNF. Given a graph with n nodes, we can use $\psi(n)$ to specify properties whose decidability is NP-hard. For example, given an undirected graph $G = \langle V, E\rangle$ with $|V| = n$, let $K_G = \langle V, E', v\rangle$, where $E' = E \cup \{\langle v, v\rangle : v \in V\}$, and v is an arbitrary node in V, and let

$$\varphi = \exists p_1 \ldots p_n[\psi(n) \wedge p_1 \wedge EX(p_2 \wedge EX(p_3 \wedge \ldots \wedge EX(p_{n-1} \wedge EX(p_n \wedge EXp_1)) \cdots))].$$

It is easy to see that both K_G and φ are of size polynomial in the size of G and that $K_G \models \varphi$ iff there exists an Hamiltonian circle in G. □

Theorem 4.3 implies that it is the modality of CTL, by itself, that makes EQCTL model checking NP-hard. Still, proving the lower bounds in the theorems above, we reduce hard problems to model checking of formulas in which the number of quantified propositions is linear in the size of the reduced problem. Thus, there is still a hope that if we restrict EQCTL and EQCTL_t to have a fixed number of quantified propositions, we get easier logics. The theorems below refute this hope. For $i \geq 0$ and $j \geq 0$, let (i,j)-EQCTL denote the restricted subset of EQCTL in which only i quantified propositions and j free propositions are allowed, and similarly for EQCTL_t.

Theorem 4.4 *The model-checking problem for* $(1,0)$*-EQCTL is NP-hard.*

Proof: We reduce SAT to $(1,0)$-EQCTL model checking. Intuitively, we do something similar to what we did for proving that EQCTL model checking is NP-hard. Since, however, a propositional formula ξ may talk about more than one proposition, we translate a formula $\xi(p_0,\ldots,p_{n-1})$ into a CTL formula that instead of talking about the value of p_i in the initial state, talks about the value of a single atomic proposition q in a state located i positions from the initial state. Formally, for $n \geq 1$, let K^n be the frame $\langle\{0,\ldots,n-1\},R,0\rangle$ where $R = \{\langle 0,1\rangle,\langle 1,2\rangle,\ldots,\langle n-2,n-1\rangle,\langle n-1,0\rangle\}$. The frame K^5 is presented in Figure 1. Giving a propositional formula ξ over $p_0,\ldots,p_{n-1}$, let ψ be the CTL formula obtained from replacing each occurrence of p_i in ξ by $(EX)^i q$. For example, if $\xi = (p_0 \vee p_1) \wedge (\neg p_1 \vee p_2)$, then $\psi = (q \vee EXq) \wedge (\neg EXq \vee EXEXq)$. It is easy to see that ξ is satisfiable iff $K^n \models \exists q \psi$. $\square$

Note that constructing ψ above, we needed a fragment of $(1,0)$-EQCTL that contains the temporal operator EX only. The satisfiability problem for this fragement can be solved in linear time. Nevertheless, model-checking complexity of this fragement is NP-hard. Thus, there are branching temporal logics with existential quantification for which model checking is harder than satisfiability.

Theorem 4.5 *The model-checking problem for* $(1,1)$*-*$EQCTL_t$ *is EXPTIME-hard.*

Proof: We reduce satisfiability of CTL to $(1,1)$-EQCTL_t model checking. Typically, we do something similar to what we did for proving that EQCTL_t model checking is EXPTIME-hard. Yet, as here we have only a single quantified proposition, we have to encode the states of K_m, as we did for the initial state in the proof of Theorem 4.4. Given $m \geq 1$ and $n \geq 1$, let $K_m^n = \langle\{start\},W,R,w^0,L\rangle$ be the Kripke structure defined as follows:

- $W = \{1,\ldots m\} \times \{0,\ldots,n-1\}$.
- $R = \{\langle (i,n-1),(k,0)\rangle, \langle (i,j),(i,j+1)\rangle : 1 \leq i,k \leq m, 0 \leq j \leq n-2\}$.
- $w^0 = (1,0)$.
- For all $1 \leq i \leq m$, we have $L((i,0)) = \{start\}$ and $L((i,j)) = \emptyset$ for all $j \neq 0$.

The frame of K_4^5 is presented in Figure 1. Now we have to translate a CTL formula $\psi(p_0, \ldots, p_{n-1})$ into a formula that instead of talking about the value of p_j at a state i of K_m, talks about the value of q at the state located j positions after the state $(i, 0)$ in K_m^n. For example, the formula $EF(p_j \wedge AGp_i)$ is translated to the formula

$$EF(start \wedge (EX)^j q \wedge AG(start \rightarrow (EX)^i q)).$$

Such a translation may increase the formula ψ by at most a factor of $|\psi|$ (because of the extra EX's). Formally, we present a function f such that ψ of length m over $p_0 \ldots p_{n-1}$ is satisfiable iff $\exists q f(\psi)$ is satisfied in K_m^n. We define f by induction on the structure of ψ as follows (Q stands for either E or A):

- $f(p_i) = (EX)^i q$.
- $f(\neg\psi_1) = \neg f(\psi_1)$.
- $f(\psi_1 \vee \psi_2) = f(\psi_1) \vee f(\psi_2)$.
- $f(QX\psi_1) = (QX)^m f(\psi_1)$.
- $f(Q\psi_1 U \psi_2) = Q(strat \rightarrow f(\psi_1))U(start \wedge f(\psi_2))$.

Note that the definition of K_m^n guarantees that path quantification in $f(\psi)$ plays a role only when interpreted in states $\{1, \ldots, m\} \times \{n-1\}$. $\square$

In fact, a more sophisticated construction can avoid the free proposition *start* (e.g., by encoding the beginning of a sequence which encodes the assignment to the atomic propositions by a sequence that does not appear elsewhere), thus showing that the EXPTIME lower bound holds even for $(1, 0)$-EQCTL$_t$.

We have seen that the model-checking problem for EQCTL$_t$ and EQCTL$_t^\star$ is as hard as the satisfiability problem for CTL and CTL*, respectively. We now show that existential quantification does not harm satisfiability complexity, for both semantics.

Theorem 4.6

(1) *The satisfiability problem for EQCTL and EQCTL$_t$ is EXPTIME-complete.*

(2) *The satisfiability problem for EQCTL* and EQCTL$_t^\star$ is 2EXPTIME-complete.*

Proof: (1) Hardness in EXPTIME follows from hardness of the satisfiability problem for CTL. To prove membership in EXPTIME, we reduce satisfiability of a formula $\varphi = \exists p_1 \ldots p_n \psi$ to the satisfiability of the CTL formula ψ. This is straightforward for φ in EQCTL, but requires some attention for φ in EQCTL$_t$. Then, while satisfaction of ψ is checked with respect to Kripke structures, satisfaction of φ is checked with respect to computation trees. It is easy to see that if ψ is satisfiable then φ is satisfiable too. For the second direction, we need the finite model property of CTL. The proof of (2) is similar, using the 2EXPTIME bounds for CTL* [26, 11, 9]. $\square$

5 Program Complexity of Model Checking

In the previous section, we presented some cheerless results concerning the model-checking complexity of branching temporal logics augmented with existential quantification over atomic propositions. In this section we consider the program complexity of model checking for these logics.

Theorem 5.1

(1) *[14] The program complexity of EQCTL model checking is NP-complete.*

(2) *The program complexity of EQCTL* model checking is NP-complete.*

Proof: (1) Membership in NP is immediate. In [14], Halpern and Kapron reduce satisfiability of CNF formulas to model checking of a fixed formula φ in $\Sigma^1_1(\exists x\mathrm{MDL})$. Whatever the logic $\Sigma^1_1(\exists x\mathrm{MDL})$ is[1], the formula φ is equivalent to an EQCTL formula. This establishes hardness in NP.

(2) Hardness in NP follows from the hardness for EQCTL. We prove membership in NP. In order to check whether a Kripke structure K satisfies an EQCTL* formula $\exists p_1 \ldots p_n \psi$, we guess a Kripke structure K' that differs from K in at most the labeling of $p_1 \ldots p_n$. As the program complexity of CTL* model checking is in P, the result follows. $\square$

Thus, as long as we are interesting in the structure semantics, fixing the formula brings no good news. Moreover, the fact that the program complexity of EQCTL* model checking is NP-hard implies that the PSPACE complexity we have for EQCTL* model checking is practically worse than the PSPACE complexity for CTL* model checking. Indeed, while the time complexity of the first is exponential in the Kripke structure, we have that the time complexity of the latter is exponential in the formula. Fortunately, the tree semantics (rather than the structure semantics) corresponds to the natural way branching temporal logics have been used to represent computations. There, as follows from the theorem below, the time complexity is polynomial in the Kripke structure.

Theorem 5.2 *The program complexity of both EQCTL$_t$ and EQCTL$^\star_t$ is P-complete.*

Proof: Since the algorithms given in the proof of Theorem 4.2 are polynomial in the size of K, membership in P is immediate. We prove hardness in P by reducing the Alternating Graph Accessibility problem, proved to be P-complete in [15, 5], to model checking of a fixed EQCTL$_t$ formula. In the Alternating Graph Accessibility problem, we are given a directed graph $G = \langle V, E \rangle$, a partition $\mathcal{E} \cup \mathcal{U}$ of V, and two designated vertices s and t. The problem is whether *alternating_path*(s, t) is true, where *alternating_path*(x, y) holds if and only if:

[1]The logic $\Sigma^1_1(\exists x\mathrm{MDL})$ consists of formulas of the form $\exists P \exists x \psi$ where ψ is a first order formula that arises as the translation of a modal formula with unary predicates in P and binary predicate R.

1. $x = y$, or

2. $x \in \mathcal{E}$ and there exists z such that $\langle x, z \rangle \in E$ and *alternating_path*(z, y), or

3. $x \in \mathcal{U}$ and for all z such that $\langle x, z \rangle \in E$, we have *alternating_path*(z, y).

Given $G, \mathcal{E}, \mathcal{U}, s$, and t, we define $K_G = \langle \{t, exist,\ univ\}, V, E, s, L \rangle$, where for all $w \in \mathcal{E} \setminus \{t\}$, we have $L(w) = \{exists\}$, for all $w \in \mathcal{U} \setminus \{t\}$, we have $L(w) = \{univ\}$, and $L(t) = \{t\}$. Consider the fixed formula

$$\varphi = \exists q[q \wedge AG(q \rightarrow (t \vee (exist \wedge EXq) \vee (univ \wedge AXq))) \wedge AF\neg q].$$

The two leftmost conjunctions in φ label with q nodes of $\langle T_{K_G}, V_{K_G} \rangle$ that correspond to states $z \in V$ for which *alternating_path*(z, t) should still be verified in order to guarantee that *alternating_path*(s, t) holds. Since φ also requires that eventually no such z is left, we have that *alternating_path*(s, t) holds iff $K_G \models_t \varphi$. Note that, as with $G2(p)$, the formula φ is not appropriate for the structure semantics. $\square$

Acknowledgment: I thank Rajeev Alur and Moshe Vardi for helpful comments.

References

[1] I. Beer, S. Ben-David, D. Geist, R. Gewirtzman, and M. Yoeli. Methodology and system for practical formal verification of reactive hardware. In *Proc. 6th Workshop on Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 182–193, Stanford, June 1994.

[2] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In D. L. Dill, editor, *Computer Aided Verification, Proc. 6th Int. Conference*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155, Stanford, California, June 1994. Springer-Verlag, Berlin.

[3] R.E. Bryant. Graph-based algorithms for boolean-function manipulation. *IEEE Trans. on Computers*, C-35(8), 1986.

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

[5] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, January 1981.

[6] 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.

[7] E.M. Clarke, D.E. Long, and K.L. McMillan. Compositional model checking. In *Proc. 4th IEEE Symposium on Logic in Computer Science*, pages 353–362, 1989.

[8] C. Courcoubetis, M.Y. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1:275–288, 1992.

[9] E.A. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, White Plains, October 1988.

[10] E.A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. In *Proceedings of the Twelfth ACM Symposium on Principles of Programming Languages*, pages 84–96, New Orleans, January 1985.

[11] E.A. Emerson and A. P. Sistla. Deciding branching time logic. In *Proceedings of the 16th ACM Symposium on Theory of Computing*, Washington, April 1984.

[12] M.J. Fischer and R.E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and Systems Sciences*, 18:194–211, 1979.

[13] O. Grumberg and D.E. Long. Model checking and modular verification. In *Proc. 2nd Conferance on Concurrency Theory*, volume 527 of *Lecture Notes in Computer Science*, pages 250–265. Springer-Verlag, 1991.

[14] J.Y. Halpern and B. Kapron. Zero-one laws for modal logic. *Annals of Pure and Applied Logic*, 69:157–193, 1994.

[15] N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.

[16] O. Kupferman and A. Pnueli. Once and for all. In *Proc. 10th IEEE Symposium on Logic in Computer Science*, San Diego, June 1995.

[17] L. Lamport. Sometimes is sometimes "not never" - on the temporal logic of programs. In *Proceedings of the 7th ACM Symposium on Principles of Programming Languages*, pages 174–185, January 1980.

[18] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of the Twelfth ACM Symposium on Principles of Programming Languages*, pages 97–107, New Orleans, January 1985.

[19] D.E. Long. *Model checking, abstraction and compositional verification.* PhD thesis, Carnegie-Mellon University, Pittsburgh, 1993.

[20] S. Pinter and P. Wolper. A temporal logic for reasoning about partially ordered computations. In *Proc. 3rd ACM Symposium on Principles of Distributed Computing*, pages 28–37, Vancouver, August 1984.

[21] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.

[22] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the Sixteenth ACM Symposium on Principles of Programming Languages*, Austin, Januery 1989.

[23] W.J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. on Computer and System Sciences*, 4:177–192, 1970.

[24] A.P. Sistla. *Theoretical issues in the design of distributed and concurrent systems.* PhD thesis, Harward University, Cambridge, MA, 1983.

[25] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[26] 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.

[27] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2):182–21, April 1986.

[28] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.

[29] P. Wolper and P. Godefroid. Partial-order methods for temporal verification. In *Proc. 4th Conferance on Concurrency Theory*, volume 715 of *Lecture Notes in Computer Science*, pages 233–246, Hildesheim, August 1993. Springer-Verlag.