CTL^+ is Complete for Double Exponential Time

Jan Johannsen and Martin Lange

Institut für Informatik
Ludwig-Maximilians-Universität München
Munich, Germany
{jjohanns,mlange}@informatik.uni-muenchen.de

Abstract. We show that the satisfiability problem for CTL^+, the branching time logic that allows boolean combinations of path formulas inside a path quantifier but no nesting of them, is 2-EXPTIME-hard. The construction is inspired by Vardi and Stockmeyer's 2-EXPTIME-hardness proof of CTL^*'s satisfiability problem. As a consequence, there is no subexponential reduction from CTL^+ to CTL which preserves satisfiability.

1 Introduction

In the early 80s, a family of branching time logics was defined by Emerson and Halpern [3, 4]. This included the commonly known logics CTL and CTL^* as well as the less known logic CTL^+.

CTL formulas can only speak about states of a transition system, while CTL^* allows properties of paths and states to be expressed. CTL^+ is the fragment of CTL^* which does not allow temporal operators to be nested. It subsumes CTL syntactically.

Emerson and Halpern [3] already showed that every CTL^+ formula is equivalent to a CTL formula. The translation, however, yields formulas of exponential length. Recently, Wilke [10] and Adler and Immerman [1] have shown that this is unavoidable, i.e. that there are CTL^+ formulas of size n such that every equivalent CTL formula is of size $\Omega(n!)$.

This gap becomes apparent for example when the complexity of the model checking problem for these logics is considered. For CTL the problem is PTIME-complete, even in linear time, while the CTL^+ model checking problem is Δ_2-complete in the polynomial time hierarchy [8].

Kupferman and Grumberg [7] have shown that one can relax the syntactic restrictions CTL imposes on branching time formulas without having to give up linear time model checking. They define a logic CTL^2, which allows two temporal operators in the scope of a path quantifier - either nested or a boolean combination thereof. Syntactically, CTL^+ and CTL^2 are incomparable although semantically CTL^2 strictly subsumes CTL and therefore CTL^+ as well. To the best of our knowledge, no complexity bounds on CTL^2's satisfiability problem are given.

In contrast, CTL^* which is known to be strictly more expressive than CTL, CTL^+ and even CTL^2, has a PSPACE-complete model checking problem [6].

Concerning the satisfiability checking problem, CTL is EXPTIME-complete while CTL^* is 2-EXPTIME-complete. Inclusion in 2-EXPTIME was proved by Emerson and Jutla [5] after it had been shown to be contained in various deterministic and nondeterministic complexity classes between 2-EXPTIME and 4-EXPTIME. 2-EXPTIME-hardness was shown by Vardi and Stockmeyer [9] using a reduction from the word problem for an alternating exponential space bounded Turing Machine.

We use the basic ideas of their construction in order to prove 2-EXPTIME-hardness of CTL^+'s satisfiability checking problem. For instance, we also encode the computation tree of an alternating exponential space bounded Turing Machine on an input word by a tree model for a CTL^+ formula that describes the machine's behaviour. However, in order to overcome CTL^+'s weaknesses in expressivity compared to CTL^* we need to make amendments to the models and the resulting formulas. Note that CTL^+ is, for example, not able to speak about the penultimate state on a finite path which is a crucial point in Vardi and Stockmeyer's reduction.

To overcome this problem we use a special type of alternating Turing Machine which is easily seen to be equivalent to a common one in terms of space complexity. This Turing Machine has states of three different types: those in which the tape head is deterministically moved, as well as existentially and universally branching states in which the symbol under the tape head is replaced and no movement takes place.

For this sort of alternating Turing Machine it becomes possible to describe the machine's behaviour by a CTL^+ formula. The distinction of Turing Machine states does not require formulas that speak about more than two consecutive states on a path of a transition system.

There are other CTL^* formulas in Vardi and Stockmeyer's paper which cannot easily be transformed into CTL^+ because of CTL^+'s restriction regarding the nesting of path operators. E.g. the natural way of expressing that some event E happens at most once along a path uses two nested *until* formulas ("it is not the case that E happens at some point and at another point later on"). Formulas of this kind occur in properties like "there is exactly one tape head per configuration". To make the reduction work for CTL^+ too, we use additional atomic propositions in a model for the resulting CTL^+ formula.

Completeness follows from the fact that the satisfiability checking problem for CTL^* is in 2-EXPTIME, but also because CTL^+ can be translated into CTL at the cost of an exponential blow-up. This does not only – to the best of our knowledge – provide the first complexity-theoretical completeness result for the CTL^+ satisfiability problem. It also shows the curious fact that concerning expressiveness CTL and CTL^+ fall into the same class different from CTL^*. Concerning the model checking problem the three logics were shown to be complete for three (probably) different classes. But regarding satisfiability, CTL$^+$ and CTL* are complete for the same class which is different from the complexity of CTL satisfiability.

Finally, we present a consequence of CTL$^+$'s 2-EXPTIME-hardness. Wilke was the first to prove an exponential lower bound on the size of CTL formulas that arise under an equivalence preserving translation from CTL$^+$ [10]. This was improved by Adler and Immerman, who showed that there is indeed an $n!$ lower bound [1]. The 2-EXPTIME-hardness of the CTL$^+$ satisfiability problem strengthens Wilke's result in a different way: there is no subexponential reduction from CTL$^+$ to CTL that preserves satisfiability.

2 Preliminaries

The logic CTL$^+$. Let $\mathcal{P}$ be a finite set of propositional constants including $\mathtt{tt}$ and $\mathtt{ff}$. A labelled transition system is a triple $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ s.t. $(\mathcal{S}, \rightarrow)$ is a directed graph, and $L : \mathcal{S} \rightarrow 2^{\mathcal{P}}$ labels the elements of $\mathcal{S}$, called *states*, with $\mathtt{tt} \in L(s), \mathtt{ff} \notin L(s)$ for all $s \in \mathcal{S}$. $\mathcal{T}$ is called *total* if for all $s \in \mathcal{S}$ there is an $s' \in \mathcal{S}$ s.t. $s \rightarrow s'$.

A path in a total transition system $\mathcal{T}$ is an infinite sequence $\pi = s_0 s_1 \ldots$ of states s.t. $s_i \rightarrow s_{i+1}$ for all $i \in \mathbb{N}$. With π^i we denote the suffix of π starting with the i-th state.

Formulas of CTL$^+$ are given by the following grammar.

$$\begin{array}{lcl} \varphi & ::= & q \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathtt{E}\psi \\ \psi & ::= & q \mid \psi \vee \psi \mid \neg\psi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \end{array}$$

where q ranges over $\mathcal{P}$. The φ are often called *state formulas* while the ψ are *path formulas*. Only state formulas are CTL$^+$ formulas. Path formulas can only occur as subformulas of these.

We will use the standard abbreviations $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \rightarrow \psi := \neg\varphi \vee \psi$, $\mathtt{A}\varphi := \neg\mathtt{E}\neg\varphi$, $\mathtt{F}\varphi := \mathtt{tt}\mathtt{U}\varphi$ and $\mathtt{G}\varphi := \neg\mathtt{F}\neg\varphi$. Furthermore, we will use a special until formula $\mathtt{F}_\psi\varphi := \neg\psi\mathtt{U}(\psi \wedge \varphi)$ which says that eventually φ holds in the first moment when ψ holds, too.

Formulas of CTL$^+$ are interpreted over paths $\pi = s_0 s_1 \ldots$ of a total transition system $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$.

$$\begin{array}{lll} \pi \models q & \text{iff} & q \in L(s_0) \\ \pi \models \varphi \vee \psi & \text{iff} & \pi \models \varphi \text{ or } \pi \models \psi \\ \pi \models \neg\varphi & \text{iff} & \pi \not\models \varphi \\ \pi \models \mathtt{E}\varphi & \text{iff} & \exists \pi',\ \text{s.t. } \pi' = s_0 \ldots \text{ and } \pi' \models \varphi \\ \pi \models \mathtt{X}\varphi & \text{iff} & \pi^1 \models \varphi \\ \pi \models \varphi\mathtt{U}\psi & \text{iff} & \exists k \in \mathbb{N} \text{ s.t. } \pi^k \models \psi \text{ and } \forall i < k : \pi^i \models \varphi \end{array}$$

Since the truth value of a state formula φ in a path $\pi = s_0 s_1 \ldots$ only depends on s_0, it is possible to write $s \models \varphi$ for a state s of a transition system and such a formula φ. A state formula φ is called *satisfiable* if there is a transition system $\mathcal{T}$ with a state s, s.t. $s \models \varphi$.

Alternating Turing Machines. We use the following model of alternating Turing Machine, which differs slightly from the standard model [2], but is easily seen to be equivalent w.r.t. space complexity. An alternating Turing Machine $\mathcal{M}$ is of the form $\mathcal{M} = (Q, \Sigma, q_0, q_a, q_r, \delta)$, where Q is the set of states, Σ is the alphabet, which contains a blank symbol $\Box \in \Sigma$, and $q_0, q_a, q_r \in Q$.

The set Q of states is partitioned into $Q = Q_\exists \cup Q_\forall \cup Q_m \cup \{q_a, q_r\}$, where we write Q_b for $Q_\exists \cup Q_\forall$, these are the *branching* states. The transition relation δ is of the form

$$\delta \subseteq \big(Q_b \times \Sigma \times Q \times \Sigma\big) \cup \big(Q_m \times \Sigma \times Q \times \{L, R\}\big) .$$

In a branching state $q \in Q_b$, the machine can act nondeterministically and writes on the tape, i.e., for each $a \in \Sigma$, there can be several transitions $(q, a, q', b) \in \delta$ for $q' \in Q$ and $b \in \Sigma$, meaning that the machine overwrites the a in the current tape cell with b, the machine enters state q', and the head does not move.

In a state $q \in Q_m$, the machine acts deterministically and moves its head, i.e., for each $a \in \Sigma$, there is exactly one transition $(q, a, q', D) \in \delta$, for $q' \in Q$ and $D \in \{L, R\}$, meaning that the head moves to the left (L) or right (R), and the machine enters state q'. For $q \in \{q_a, q_r\}$, there are no transitions in δ, and the machine halts.

We assume that the machine only halts when the state is q_a or q_r. A halting configuration is accepting iff the state is q_a. For the other configurations, the acceptance behaviour depends on the kind of state:

If the state is in Q_m, then the configuration is accepting iff its unique successor is accepting. If the state is in $Q_\exists$, then the configuration is accepting iff at least one of its successors is accepting. If the state is in $Q_\forall$, then the configuration is accepting iff all of its successors are accepting. The whole computation accepts if the initial configuration is accepting.

Double exponential time. The complexity class of double exponential time is defined as

$$\text{2-EXPTIME} = \bigcup_{k \in \mathbb{N}} \text{DTIME}(2^{2^{k \cdot n}})$$

where DTIME(f(n)) is the class of all languages which are accepted by a deterministic Turing Machine in time $f(n)$ where n is the length of the input word at hand.

It is well-known [2] that 2-EXPTIME coincides with

$$\text{AEXPSPACE} = \bigcup_{k \in \mathbb{N}} \text{ASPACE}(2^{k \cdot n})$$

the class of all languages accepted by an alternating Turing Machine using space which is at most exponential in the size of the input word.

3 The Reduction

Theorem 1. *Satisfiability of CTL$^+$ is 2-EXPTIME-hard.*

Proof. Suppose $\mathcal{M} = (Q, \Sigma, q_0, q_a, q_r, \delta)$ is an alternating exponential space bounded Turing Machine. Let $w = a_0 \ldots a_{n-1} \in \Sigma^*$ be an input for $\mathcal{M}$. W.l.o.g. we assume the space needed by $\mathcal{M}$ on input w to be bounded by $2^{kn} - 1$ for some $k \geq 1$. Let $N := 2^{kn} - 1$. Furthermore we assume that every computation ends in a configuration with the head on the rightmost tape cell while the machine is in either of he states q_a or q_r.

In the following we will construct a CTL$^+$ formula $\varphi_{\mathcal{M},w}$ s.t. $w \in L(\mathcal{M})$ iff $\varphi_{\mathcal{M},w}$ is satisfiable. Informally, an accepting computation of $\mathcal{M}$ on w will serve as a model for $\varphi_{\mathcal{M},w}$.

Like Vardi and Stockmeyer [9], we encode a configuration of $\mathcal{M}$ as a sequence of $2^{k \cdot n} - 1$ states in a possible model for $\varphi_{\mathcal{M},w}$. Successive configurations of the Turing Machine are modelled by concatenating these sequences, where we add one dummy state with index 0 between each pair of adjacent configurations.

The underlying set of propositions is $\mathcal{P} = Q \cup \Sigma \cup \{c_0, \ldots, c_{k \cdot n - 1}\} \cup \{x, z, e\}$.

- $q \in Q$ is true in a state of the model iff the head of the Turing Machine is on the corresponding tape cell in the corresponding configuration while the machine is in state q. The formula $h := \bigvee_{q \in Q} q$ says that the machine is in some state, i.e. the head is on that cell.
- $a \in \Sigma$ is true iff a is the symbol on the corresponding tape cell.
- $c_{k \cdot n - 1}, \ldots, c_0$ represent a counter in binary representation. The counter value in a state of the model is 0 at the dummy states and the number of the corresponding tape cell otherwise.
- x is used to denote that the corresponding configuration is accepting.
- z is used to mark the part of a tree model which corresponds to the computation. In order to be able to speak about a certain state somewhere on a path we let every state of the encoding have a successor which carries exatly the same amount of information except that it is labelled with $\neg z$. Thus, such a state can be seen as not belonging directly to the encoding of the computation tree but being a clone of a state in this tree.
- e indicates that the state at hand belongs to an "even" configuration, i.e. one with an even index in a sequence $C_0, C_1, \ldots$ of configurations of the computation.

For every fixed m we can write a formula χ_m which says that the counter value is m in the current state, e.g.

$$\chi_0 := \bigwedge_{i=0}^{k \cdot n - 1} \neg c_i \ , \ \chi_1 := c_0 \wedge \bigwedge_{i=1}^{k \cdot n - 1} \neg c_i \text{ and } \chi_N := \bigwedge_{i=0}^{k \cdot n - 1} c_i$$

for the dummy ($m = 0$), the leftmost ($m = 1$) and rightmost ($m = N$) position in a configuration.

In order to describe $\mathcal{M}'s$ behaviour on w we need to express several properties. The formula φ_0 says that there is always exactly one symbol on a tape cell, and $\mathcal{M}$ is never in two different states at the same time.

$$\varphi_0 \;:=\; \mathtt{AG}(\; (\neg\chi_0 \;\rightarrow\; \bigvee_{a\in\Sigma} a) \;\wedge\; (\chi_0 \;\rightarrow\; \neg h \wedge \bigwedge_{a\in\Sigma} \neg a) \;\wedge$$
$$\bigwedge_{a,b\in\Sigma,b\neq a} \neg(a\wedge b) \;\wedge\; \bigwedge_{q,q'\in Q,q\neq q'} \neg(q\wedge q')\;)$$

We can say that the counter value is not changed in the transition to the next state on a given path. This is used to clone states as indicated above. The value of e does not change in this case.

$$\psi_{rem} \;:=\; (e \leftrightarrow \mathtt{X}e) \;\wedge\; \bigwedge_{j=0}^{k\cdot n-1} (\; c_j \leftrightarrow \mathtt{X}c_j\;)$$

We can also say that the counter value is increased by 1 modulo $2^{k\cdot n}$. Then, a switch from e to $\neg e$ or vice versa occurs iff the counter is increased from $2^{k\cdot n}-1$ to 0.

$$\psi_{inc} \;:=\; (\; (e \leftrightarrow \mathtt{X}\neg e) \;\wedge\; \chi_N \wedge \mathtt{X}\chi_0\;) \vee$$
$$(e \leftrightarrow \mathtt{X}e) \;\wedge\; \bigvee_{j=0}^{k\cdot n-1} (\; \neg c_j \wedge \mathtt{X}c_j \wedge \bigwedge_{i>j}(c_i \leftrightarrow \mathtt{X}c_i) \;\wedge\; \bigwedge_{i<j}(c_i \wedge \mathtt{X}\neg c_i)\;)$$

The entire computation of $\mathcal{M}$ forms a tree. Each state is labelled with a symbol of Σ. Moreover, z holds on every state on the computation, and every state has at least one successor from which on z never holds. Furthermore, the subtree under this state reflects the labelling of its root's predecessor which still satisfies z. This idea is taken from Vardi and Stockmeyer's proof [9] and used to be able to speak about finite prefixes of infinite paths.

On all paths q_a or q_r is eventually reached and all following states do not satisfy z. The counter is only increased (modulo $2^{k\cdot n}$) in states satisfying z.

$$\begin{aligned}
\psi_{eq} \;&:=\; \psi_{rem} \;\wedge\; \bigwedge_{q\in Q} q \leftrightarrow \mathtt{X}q \;\wedge\; \bigwedge_{a\in\Sigma} a \leftrightarrow \mathtt{X}a \\
\varphi_1 \;&:=\; \mathtt{AF}\neg z \;\wedge \\
&\quad\; \mathtt{AG}(\; (z \wedge \neg q_a \wedge \neg q_r) \;\rightarrow\; (\mathtt{EX}z \wedge \mathtt{EX}\neg z) \;\wedge \\
&\qquad\quad \neg z \;\rightarrow\; \mathtt{A}(\mathtt{X}\neg z \wedge \psi_{eq}) \;\wedge \\
&\qquad\quad (q_a \vee q_r) \;\rightarrow\; \mathtt{AX}\neg z \wedge \chi_N\;) \;\wedge \\
&\quad\; \mathtt{AGA}(\; (z \wedge \mathtt{X}z \;\leftrightarrow\; \psi_{inc}) \;\wedge\; (z \wedge \mathtt{X}\neg z \;\leftrightarrow\; \psi_{eq})\;)
\end{aligned}$$

There is at most one tape head in every configuration. (The fact that there is at least one will be guaranteed by φ_5 later on.) This is achieved by saying that there is no bit c_i which distinguishes two possible occurrences of an h in one configuration. To guarantee that one speaks about the same configuration for two such occurrences of h, we demand that the value of e never changes in between.

$$\begin{aligned}\varphi_2 \;:=\; \texttt{AGA}(\; \chi_0 \to (e \to \neg(\bigvee_{i=0}^{k\cdot n-1} e\texttt{U}(e \wedge h \wedge c_i) \wedge e\texttt{U}(e \wedge h \wedge \neg c_i)) \wedge \\ \neg e \to \neg(\bigvee_{i=0}^{k\cdot n-1} \neg e\texttt{U}(\neg e \wedge h \wedge c_i) \wedge \neg e\texttt{U}(\neg e \wedge h \wedge \neg c_i))\;)\;)\end{aligned}$$

The computation is accepting. Every q_a is marked with an x but no q_r is. Moreover, an x occurs together with an existential state only if there is a path along z s.t. x holds together with the first occurrence of h. For universal or moving states all z-paths must satisfy x in their first occurrence of h.

$$\begin{aligned}\varphi_3 \;:=\; x \;\wedge\; \texttt{AG}(\; (q_a \to x) \;\wedge\; (q_r \to \neg x) \;\wedge \\ \bigwedge_{q \in Q_\exists} q \;\to\; (x \leftrightarrow \texttt{EXE}((z \wedge \neg h)\texttt{U}(z \wedge h \wedge x))) \;\wedge \\ \bigwedge_{q \in Q_\forall \cup Q_m} q \;\to\; (x \leftrightarrow \texttt{AXA}(z\texttt{U}(z \wedge h) \to \texttt{F}_h x))\;)\end{aligned}$$

At the beginning, the tape contains $a_1 \ldots a_n \Box \ldots \Box$, the input word followed by $2^{k\cdot n} - n$ blank symbols. $\mathcal{M}$ is in state q_0 and the head is on the first symbol of w.

$$\begin{aligned}\varphi_4 \;:=\; & z \;\wedge\; e \;\wedge\; \chi_0 \;\wedge \\ & \texttt{EX}(\; z \;\wedge\; q_0 \;\wedge\; a_1 \;\wedge \\ & \quad \texttt{EX}(\; z \;\wedge\; a_2 \;\wedge \\ & \quad\quad \ldots \;\wedge \\ & \quad\quad\quad \texttt{EX}(\; z \;\wedge\; a_n \wedge \\ & \quad\quad\quad\quad \texttt{EXE}(\; (z \wedge \Box)\texttt{U}(z \wedge \chi_0)\;))\ldots))\end{aligned}$$

Now we have to say that two adjacent configurations comply with $\mathcal{M}$'s transition rules. In order to do so we need the following statements about a path. The counter value is 0 exactly once before $\neg z$ holds.

$$\begin{aligned}\psi_1 \;:=\; & e \to z\texttt{U}(z \wedge \neg e \wedge \chi_0) \;\wedge\; \neg e \to z\texttt{U}(z \wedge e \wedge \chi_0) \;\wedge \\ & \neg(\; z\texttt{U}(e \wedge \chi_0) \;\wedge\; z\texttt{U}(\neg e \wedge \chi_0)\;)\end{aligned}$$

We need three formulas saying that the counter value in the first state not satisfying z is the same as the value of the first state on the path, resp. increased or decreased by 1. We explicitly forbid to increase a maximal value, resp. decrease a minimal one, i.e. do not calculate modulo $2^{k\cdot n}$, because these formulas are used to describe the tape head's moves. Note that it cannot go left at the right end of the tape and vice-versa.

$$\psi_{=} \;:=\; \bigwedge_{i=0}^{k\cdot n-1} c_i \leftrightarrow \mathtt{F}_{\neg z} c_i$$

$$\psi_{+1} := \neg\chi_N \;\wedge\; \bigvee_{j=0}^{k\cdot n-1} (\neg c_j \wedge \mathtt{F}_{\neg z} c_j) \;\wedge\; \bigwedge_{i>j} (c_i \leftrightarrow \mathtt{F}_{\neg z} c_i) \;\wedge\; \bigwedge_{i<j} (c_i \wedge \mathtt{F}_{\neg z} \neg c_i)$$

$$\psi_{-1} := \neg\chi_1 \;\wedge\; \bigvee_{j=0}^{k\cdot n-1} (c_j \wedge \mathtt{F}_{\neg z} \neg c_j) \;\wedge\; \bigwedge_{i>j} (c_i \leftrightarrow \mathtt{F}_{\neg z} c_i) \;\wedge\; \bigwedge_{i<j} (\neg c_i \wedge \mathtt{F}_{\neg z} c_i)$$

Finally, we have to describe the machine's transition behaviour δ. On every state the following holds.

- If it is labelled with a $q \in Q_b$ then the actual symbol is replaced in every next configuration at the same position.
- If it is not labelled with a $q \in Q_b$, in particular no q at all, then the corresponding state of the next configuration carries the same symbol from Σ.
- If it is labelled with a $q \in Q_m$ then every next or previous state to the corresponding one in the next configuration is labelled with the machine state that is given by the transition relation.

Note that the second and third case do not exclude each other.

$$\begin{aligned}
\varphi_5 \;:=\; \mathtt{AG}\Big(& \bigwedge_{q\in Q_b, a\in\Sigma} \big(q \wedge a \;\rightarrow \bigwedge_{(q,a,q',b)\in\delta} \mathtt{E}(\; \psi_1 \wedge \psi_{=} \wedge \mathtt{F}_{\neg z}(q' \wedge b)\;) \;\wedge \\
& \qquad\qquad\qquad \mathtt{A}(\; \psi_1 \wedge \psi_{=} \;\rightarrow\; \mathtt{F}_{\neg z} \bigvee_{(q,a,q',b)\in\delta} (q' \wedge b)\;)\;\big) \\
& \wedge \bigwedge_{a\in\Sigma} \neg(\bigvee_{q\in Q_b} q) \wedge a \;\rightarrow\; \mathtt{A}(\; \psi_1 \wedge \psi_{=} \;\rightarrow\; \mathtt{F}_{\neg z} a\;) \\
& \wedge \bigwedge_{(q,a,q',L)\in\delta} q \wedge a \;\rightarrow\; \mathtt{A}(\; \psi_1 \wedge \psi_{-1} \;\rightarrow\; \mathtt{F}_{\neg z} q'\;) \\
& \wedge \bigwedge_{(q,a,q',R)\in\delta} q \wedge a \;\rightarrow\; \mathtt{A}(\; \psi_1 \wedge \psi_{+1} \;\rightarrow\; \mathtt{F}_{\neg z} q'\;)\Big)
\end{aligned}$$

Altogether, the machine's behaviour is described by the formula

$$\varphi_{\mathcal{M},w} \;:=\; \varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \varphi_5$$

Then, the part of a model for $\varphi_{\mathcal{M},w}$ that is marked with z corresponds to a successful computation tree of $\mathcal{M}$ on w. Conversely, such a tree can easily be extended to a model for $\varphi_{\mathcal{M},w}$.

Thus, $\mathcal{M}$ accepts w iff there exists a successful computation tree for $\mathcal{M}$ on w iff there exists a model for $\varphi_{\mathcal{M},w}$ iff $\varphi_{\mathcal{M},w}$ is satisfiable.

Finally, the size of $\varphi_{\mathcal{M},w}$ is quadratic in $|\Sigma|$ and $|Q|$ and linear in $|w|$ and $|\delta|$. $\square$

Corollary 1. *There is no reduction $r : CTL^+ \rightarrow CTL$ s.t. for all $\varphi \in CTL^+$:*

- *φ is satisfiable iff $r(\varphi)$ is satisfiable, and*

– $|r(\varphi)| \leq f(|\varphi|)$ *for some* $f : \mathbb{N} \to \mathbb{N}$ *with* $f(n^2) = o(2^n)$.

Proof. Suppose there is a reduction from CTL$^+$ to CTL that preserves satisfiability and produces formulas of subexponential length $f(n)$. Then this reduction in conjunction with a satisfiability checker for CTL can be used to decide satisfiability of CTL$^+$ in asymptotically less time than $O(2^{f(n)})$. As a consequence of Theorem 1, every language in 2-EXPTIME can be decided in time $O(2^{f(n^2)})$ since it can be reduced to CTL$^+$ in quadratic time, and satisfiability for CTL can be decided in time $O(2^n)$. But according to the asymptotic restriction on f and the Time Hierarchy Theorem, there is a language in 2-EXPTIME which is not decidable in time $O(2^{f(n^2)})$. To see this note that

$$f(n^2) = o(2^n) \text{ iff } f(n^2) + \log f(n^2) = o(2^n) \text{ iff } 2^{f(n^2)} \cdot f(n^2) = o(2^{2^n})$$

□

References

1. M. Adler and N. Immerman. An $n!$ lower bound on formula size. In *Proc. 16th Symp. on Logic in Computer Science, LICS'01*, pages 197–208, Boston, MA, USA, June 2001. IEEE Computer Society.
2. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.
3. E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.
4. E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, January 1986.
5. E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, 29(1):132–158, February 2000.
6. E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.
7. O. Kupferman and O. Grumberg. Buy one, get one free!!! *Journal of Logic and Computation*, 6(4):523–539, August 1996.
8. F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking CTL^+ and $FCTL$ is hard. In *Proc. 4th Conf. Foundations of Software Science and Computation Structures, FOSSACS'01*, volume 2030 of *LNCS*, pages 318–331, Genova, Italy, April 2001. Springer.
9. M. Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *Proc. 17th Symp. on Theory of Computing, STOC'85*, pages 240–251, Baltimore, USA, May 1985. ACM.
10. T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In *Proc. 19th Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'99*, volume 1738 of *LNCS*, pages 110–121. Springer, 1999.