CTL^+ Is Exponentially More Succinct than CTL

Thomas Wilke

Lehrstuhl für Informatik VII,
RWTH Aachen, 52056 Aachen, Germany
wilke@informatik.rwth-aachen.de

Abstract. It is proved that CTL^+ is exponentially more succinct than CTL. More precisely, it is shown that every CTL formula (and every modal μ-calculus formula) equivalent to the CTL^+ formula

$$\mathsf{E}(\mathsf{F}p_0 \wedge \cdots \wedge \mathsf{F}p_{n-1})$$

is of length at least $\binom{n}{\lceil n/2 \rceil}$, which is $\Omega(2^n/\sqrt{n})$. This matches almost the upper bound provided by Emerson and Halpern, which says that for every CTL^+ formula of length n there exists an equivalent CTL formula of length at most $2^{n \log n}$.
It follows that the exponential blow-up as incurred in known conversions of nondeterministic Büchi word automata into alternation-free μ-calculus formulas is unavoidable. This answers a question posed by Kupferman and Vardi.
The proof of the above lower bound exploits the fact that for every CTL (μ-calculus) formula there exists an equivalent alternating tree automaton of linear size. The core of this proof is an involved cut-and-paste argument for alternating tree automata.

1 Introduction

Expressiveness and *succinctness* are two important aspects to consider when one investigates a (specification) logic. When studying the expressiveness of a logic one is interested in characterizing *what* properties can be expressed, whereas when studying the succinctness one is interested in *how short* a formula can be found to express a given property. Succinctness is especially of importance in a situation where one has characterized the expressive power of a logic by a different but equally expressive logic. In such a situation, succinctness is the foremost quantitative measure to distinguish the logics. For instance, linear-time temporal logic (LTL) is known to be exactly as expressive as first-order logic (FO), [9], but FO is much more succinct than LTL: from work by Stockmeyer's, [11], it follows that there exists a sequence of FO formulas of linear length such that the length of shortest equivalent LTL formulas cannot be bounded by an elementary recursive function.

In this paper, the succinctness of computation tree logic (CTL) is compared to the succinctness of CTL^+, an extension of CTL, which is known to have exactly the same expressive power as CTL, [4,5]. I present a sequence of CTL^+

C. Pandu Rangan, V. Raman, R. Ramanujam (Eds.): FSTTCS'99, LNCS 1738, pp. 110–121, 1999.

formulas of length $\mathcal{O}(n)$ such that the length of shortest equivalent CTL formulas is $\Omega(2^n/\sqrt{n})$. More precisely, I prove that every CTL formula equivalent to the CTL$^+$ formula

$$\mathsf{E}(\mathsf{F}p_0 \wedge \cdots \wedge \mathsf{F}p_{n-1})$$

is of length at least $\binom{n}{\lceil n/2 \rceil}$, which shows that CTL$^+$ is exponentially more succinct than CTL. This lower bound is almost tight, because a result by Emerson and Halpern's, [4,5], says that for every CTL$^+$ formula of length n there exists an equivalent CTL formula of length at most $2^{n \log n}$.

It is important to note that this exponential lower bound is not based on any complexity-theoretic assumption, and it does not follow from the fact that model checking for CTL is known to be P-complete whereas model checking for CTL$^+$ is NP- and co-NP-hard (and in Δ_2^p), [3,4,5].

The proof of the lower bound presented in this paper makes use of automata-theoretic arguments, following other approaches to similar questions. The main idea is based on the following fact. For every CTL formula (and for every μ-calculus formula) φ there exists an alternating tree automaton $\boldsymbol{A}_\varphi$ of size linear in the length of φ that accepts exactly the models of φ, [6,1,2]. So in order to obtain a lower bound on the length of the CTL (or μ-calculus) formulas defining a given class of Kripke structures,[1] it is enough to establish a lower bound on the number of states of the alternating tree automata recognizing the given class of structures.

As mentioned above, automata-theoretic arguments have been used in this way in different places, for instance by Etessami, Vardi, and myself in [8] or Kupferman and Vardi in [10]. The difference, however, is that in this paper the automaton model (alternating automata on trees) is rather intricate compared to the automaton models used in [8] and [10] (nondeterministic automata on words and nondeterministic automata on trees, respectively).

The more elaborate argument that is needed here also answers a question raised in the paper by Kupferman and Vardi. A particular problem they consider is constructing for a given nondeterministic Büchi word automaton an alternation free μ-calculus (AFMC) formula that denotes in every Kripke structure the set of all worlds where all infinite paths originating in this world are accepted by the automaton. They show that if such a formula exists, then there is a formula of size at most exponential in the number of states of the given Büchi automaton, but they cannot give a matching lower bound. This is provided in this paper.

Outline. In Section 2, the syntax and semantics of CTL and CTL$^+$ are briefly reviewed and the main result of the paper is presented. In Section 3, alternating tree automata are briefly reviewed and subsequently, in Section 4, the succinctness problem is reduced to an automata-theoretic problem. Section 5 describes

[1] Strictly speaking, a CTL formula defines a class of pointed Kripke structures, see Section 2.

the latter in a more general setting and in Section 6 a sketch is given of the solution of this more general problem. Section 7 presents consequences, and Section 8 gives a conclusion.

This paper is an extended abstract; for details of the proofs, see the technical report [12].

Acknowledgment. I would like to thank Kousha Etessami, Martin Grohe, Neil Immerman, Christof Löding, Philippe Schnoebelen, and Moshe Y. Vardi for having discussed with me the problem addressed in this paper.

Trees and tree arithmetic. In this paper, a tree is a triple (V, E, λ) where (V, E) is a directed tree in the graph-theoretic sense and λ is a labeling function with domain V. By convention, when $\boldsymbol{T}$ denotes a tree, then V, E, and λ always denote the set of nodes, set of edges, and labeling function of $\boldsymbol{T}$. The same applies to decorations such as $\boldsymbol{T}'$, $\boldsymbol{T}^*$, $\boldsymbol{T}_i$, etc.

Let $\boldsymbol{T}$ be an arbitrary tree. A node $v' \in V$ is a *successor* of a node $v \in V$ in $\boldsymbol{T}$ if $(v, v') \in E$. The set of all successors of a node v in $\boldsymbol{T}$ is denoted by $\mathit{Scs}(\boldsymbol{T}, v)$. The set of *leaves* of a tree $\boldsymbol{T}$, that is, the set of nodes without successors, is denoted by $\mathit{Lvs}(\boldsymbol{T})$. The set of *inner nodes* is denoted by $\mathit{In}(\boldsymbol{T})$.

Given a tree $\boldsymbol{T}$ and a vertex v of $\boldsymbol{T}$, the *ancestors path,* denoted $\boldsymbol{T}{\uparrow}v$, is the unique path from the root of $\boldsymbol{T}$ to v (inclusively). The *descendants tree,* denoted $\boldsymbol{T}{\downarrow}v$, is the subgraph of $\boldsymbol{T}$ induced by all nodes reachable from v (v itself included).

I will use two kinds of concatenations for trees. When $\boldsymbol{T}$ and $\boldsymbol{T}'$ are trees and v is a node of $\boldsymbol{T}$, then $\boldsymbol{T} \cdot (v, \boldsymbol{T}')$ denotes the tree that results from $\boldsymbol{T}$ by first making an isomorphic copy of $\boldsymbol{T}'$ whose node set is disjoint from the set of nodes of $\boldsymbol{T}$ and then adding an edge from v to the root of $\boldsymbol{T}'$. Similarly, $\boldsymbol{T} \odot (v, \boldsymbol{T}')$ denotes the tree that results from $\boldsymbol{T}$ by first making an isomorphic copy of $\boldsymbol{T}'$ whose node set is disjoint from the set of nodes of $\boldsymbol{T}$ and then identifying the root of the isomorphic copy of $\boldsymbol{T}'$ with v. By convention, the node v is retained in the resulting tree (rather than the root of $\boldsymbol{T}'$) and the label of v is kept.— These two concatenation operations are extended in a straightforward way: when $\boldsymbol{T}$ is a tree and M a set of pairs $(v, \boldsymbol{T}')$, with $v \in V$ and $\boldsymbol{T}'$ an arbitrary tree, I might write $\boldsymbol{T} \cdot M$ and $\boldsymbol{T} \odot M$ to denote the result of concatenating (in the respective way) all trees from M to $\boldsymbol{T}$.

For ease in notation, when π is a finite path (a finite tree with exactly one leaf) with leaf v and $\boldsymbol{T}$ is a tree, I simply write $\pi \cdot \boldsymbol{T}$ for the tree $\pi \cdot (v, \boldsymbol{T})$ as defined above. To make things even simpler, I view strings as finite paths and vice versa. So when u is a string and $\boldsymbol{T}$ a tree, I might write $u \cdot \boldsymbol{T}$ to denote the tree which is obtained by viewing u as a path and concatenating $\boldsymbol{T}$ to it.

2 CTL, CTL$^+$, and Main Result

I start with recalling the syntax and the semantics of CTL and CTL$^+$. For technical reasons, I only define formulas in positive normal form. This is not an

essential restriction, because every CTL formula is equivalent to a CTL formula in positive normal form of the same length, and the same applies to CTL$^+$.

Let Prop $= \{p_0, p_1, p_2, \dots\}$ be an infinite supply of distinct propositional variables. The set of all *CTL$^+$ formulas* and the set of all *path formulas* are defined simultaneously as follows.

1. 0 and 1 are CTL$^+$ formulas.
2. For $p \in$ Prop, p and $\neg p$ are CTL$^+$ formulas.
3. If φ and ψ are CTL$^+$ formulas, then so are $\varphi \vee \psi$ and $\varphi \wedge \psi$.
4. Every CTL$^+$ formula is a path formula.
5. If φ and ψ are CTL$^+$ formulas, then $\mathsf{X}\varphi$, $\mathsf{U}(\varphi, \psi)$, and $\mathsf{R}(\varphi, \psi)$ are path formulas.
6. If φ and ψ are path formulas, then so are $\varphi \vee \psi$ and $\varphi \wedge \psi$.
7. If φ is a path formula, then $\mathsf{E}\varphi$ and $\mathsf{A}\varphi$ are CTL$^+$ formulas.

A CTL$^+$ formula is a *CTL formula* when it can be constructed without using rule 6. That is, in CTL formulas every path quantifier (E or A) is followed immediately by a temporal modality (X, U, or R). As usual, I use $\mathsf{F}\varphi$ (eventually φ) as an abbreviation for $\mathsf{U}(1, \varphi)$.

CTL and CTL$^+$ formulas are interpreted in Kripke structures, which are directed graphs with specific labeling functions for their nodes. Formally, a *Kripke structure* is a tuple (W, R, α) where W is a set of *worlds,* $R \subseteq W \times W$ is an *accessibility relation,* and $\alpha\colon W \to 2^{\mathrm{Prop}}$ is a *labeling function,* which assigns to each world the set of propositional variables that hold true in it. By convention, Kripke structures are denoted by $\boldsymbol{K}$ or decorated versions of $\boldsymbol{K}$ such as $\boldsymbol{K'}$ or $\boldsymbol{K^*}$, and their components are referred to as W, R, and α, respectively decorated versions of these letters.

Given a world w of a Kripke structure $\boldsymbol{K}$ as above, a world w' is called a *successor* of w in $\boldsymbol{K}$ if $(w, w') \in R$. Just as with trees, the set of all successors of a world w is denoted by $\mathit{Scs}(\boldsymbol{K}, w)$. A *path* through a Kripke structure $\boldsymbol{K}$ as above is a nonempty sequence $w_0, w_1, \dots$ such that $(w_0, w_1) \in R$, $(w_1, w_2) \in R$, $\dots$ A *maximal path* is a path that is either infinite or finite and ends in a world without successors.

A *pointed Kripke structure* is a pair $(\boldsymbol{K}, w)$ of a Kripke structure and a *distinguished* world of it. A *path* through a pointed Kripke structure $(\boldsymbol{K}, w)$ is a path through $\boldsymbol{K}$ starting in w. A *path-equipped Kripke structure* is a pair $(\boldsymbol{K}, \pi)$ of a Kripke structure and a maximal path through it.

For every CTL$^+$ and path formula φ, one defines in a straightforward way what it means for φ to hold in a pointed Kripke structure $(\boldsymbol{K}, w)$ respectively path-equipped Kripke structure $(\boldsymbol{K}, \pi)$ and denotes this by $(\boldsymbol{K}, w) \models \varphi$ respectively $(\boldsymbol{K}, \pi) \models \varphi$. For instance, when φ is a path formula, then $(\boldsymbol{K}, w) \models \mathsf{E}\varphi$ if there exists a maximal path π through $(\boldsymbol{K}, w)$ such that $(\boldsymbol{K}, \pi) \models \varphi$. For details, the reader is referred to [5].

Given a CTL$^+$ formula φ, I write $\mathrm{Mod}(\varphi)$ for the class of all pointed Kripke structures that are *models* of φ, i. e., $\mathrm{Mod}(\varphi) = \{(\boldsymbol{K}, w) \mid (\boldsymbol{K}, w) \models \varphi\}$. CTL$^+$ formulas φ and ψ are *equivalent* if they have the same models, i. e., if $\mathrm{Mod}(\varphi) = \mathrm{Mod}(\psi)$.

The main result of this paper is:

Theorem 1. *Every* CTL *formula equivalent to the* CTL$^+$ *formula* φ_n *defined by*

$$\varphi_n = \mathsf{E}(\mathsf{F}p_0 \wedge \cdots \wedge \mathsf{F}p_{n-1}) \tag{1}$$

has length at least $\binom{n}{\lceil n/2 \rceil}$*, which is* $\Omega(2^n/\sqrt{n})$*.*

In other words, CTL$^+$ *is exponentially more succinct than* CTL*.*

Note that it is easy to come up with a formula of length $\mathcal{O}(n!)$ equivalent to φ_n, namely as a disjunction with $n!$ many disjuncts, each taking care of one possible order in which the p_i's may occur on a path.

3 Alternating Tree Automata

As indicated in the abstract and the introduction, I use an automata-theoretic argument to prove Theorem 1. In this section, the respective automaton model, which differs from other models used in the literature, is introduced.

First, it can handle trees with arbitrary degree of branching in a simple way. Second, the class of objects accepted by an automaton as defined here is a class of pointed Kripke structures rather than just a set of trees. Both facts make it much easier to phrase theorems such as Theorem 2 below and also simplify the presentation of a combinatorial (lower-bound) argument like the one given in Section 6.

An *alternating tree automaton (ATA)* is a tuple $\boldsymbol{A} = (Q, P, q_I, \delta, \Omega)$ where Q is a finite set of *states*, P is a finite subset of Prop, $q_I \in Q$ is an *initial state*, δ is a *transition function* as specified below, and Ω is an *acceptance condition* for ω-automata such as a Büchi or Muller condition. The same notational conventions as with Kripke structures apply.

The transition function δ is a function $Q \times 2^P \to \mathrm{TC}(Q)$, where $\mathrm{TC}(Q)$ is the set of *transition conditions* over Q, which are defined by the following rules.

1. 0 and 1 are transition conditions over Q.
2. For every $q \in Q$, q is a transition condition over Q.
3. For every $q \in Q$, $\Box q$ and $\Diamond q$ are transition conditions over Q.
4. If φ and ψ are transition conditions over Q, then $\varphi \wedge \psi$ and $\varphi \vee \psi$ are transition conditions over Q.

A transition condition is said to be *ϵ-free* if rule 2 is not needed to build it. An ATA is ϵ-free if every condition $\delta(q, a)$ for $q \in Q$ and $a \in 2^P$ is ϵ-free; it is in *normal form* if it is ϵ-free, the conditions $\delta(q, a)$ are in disjunctive normal form, and neither 0 nor 1 occur in these conditions.

ATA's work on pointed Kripke structures. Their computational behavior is explained using the notion of a run. Assume $\boldsymbol{A}$ is an ATA as above and $(\boldsymbol{K}, w_I)$ a pointed Kripke structure as above. A *run* of $\boldsymbol{A}$ on $(\boldsymbol{K}, w_I)$ is a $(W \times Q)$-labeled tree $\boldsymbol{R} = (V, E, \lambda)$ satisfying the conditions described further below. To explain these conditions, some more definitions are needed. For simplicity

in notation, I will write $w_{\boldsymbol{R}}(v)$ and $q_{\boldsymbol{R}}(v)$ for the first and second component of $\lambda(v)$, respectively.

For every node v of $\boldsymbol{R}$, I define what it means for a transition condition τ over Q to hold in v, denoted $\boldsymbol{K}, \boldsymbol{R}, v \models \tau$. This definition is by induction on the structure of τ, where the boolean constants 0 and 1 and the boolean connectives are dealt with in the usual way; besides:

— $\boldsymbol{K}, \boldsymbol{R}, v \models q$ if there exists $v' \in Scs(\boldsymbol{R}, v)$ such that $\lambda(v') = (w_{\boldsymbol{R}}(v), q)$,

— $\boldsymbol{K}, \boldsymbol{R}, v \models \Diamond q$ if there exists $v' \in Scs(\boldsymbol{R}, v)$ and $w \in Scs(\boldsymbol{K}, w_{\boldsymbol{R}}(v))$ such that $\lambda(v') = (w, q)$, and

— $\boldsymbol{K}, \boldsymbol{R}, v \models \Box q$ if for every $w \in Scs(\boldsymbol{K}, w_{\boldsymbol{R}}(v))$ there exists $v' \in Scs(\boldsymbol{R}, v)$ such that $\lambda(v') = (w, q)$.

The two additional conditions that are required of a run are the following.

1. *Initial condition.* Let v_0 be the root of (V, E). Then $\lambda(v_0) = (w_I, q_I)$.
2. *Local consistency.* For every $v \in V$,

$$\boldsymbol{K}, \boldsymbol{R}, v \models \tau_v \tag{2}$$

where

$$\tau_v = \delta(q_{\boldsymbol{R}}(v), \alpha(w_{\boldsymbol{R}}(v)) \cap P) \ . \tag{3}$$

Note that the intersection with P allows us to deal easily with the fact that in the definition of Kripke structure an infinite number of propositional variables is always present.

A run $\boldsymbol{R}$ is said to be *accepting* if the state labeling of every infinite path through $\boldsymbol{R}$ satisfies the given *acceptance condition* Ω. For instance, if $\Omega \subseteq 2^Q$ is a Muller condition, then every infinite path $v_0, v_1, \ldots$ through $\boldsymbol{R}$ must have the property that the set formed by the states occurring infinitely often in $q_{\boldsymbol{R}}(v_0), q_{\boldsymbol{R}}(v_1), \ldots$ is a member of Ω.

A pointed Kripke structure is *accepted* by $\boldsymbol{A}$ if there exists an accepting run of $\boldsymbol{A}$ on the Kripke structure. The class of pointed Kripke structures accepted by $\boldsymbol{A}$ is denoted by $\mathcal{K}(\boldsymbol{A})$; it is said $\boldsymbol{A}$ *recognizes* $\mathcal{K}(\boldsymbol{A})$.

Throughout this paper, the same notational conventions as with Kripke structures and alternating tree automata apply to runs.

4 Reduction to Automata-Theoretic Problem

In order to reduce the lower bound claim for the translation from CTL$^+$ to CTL to a claim on alternating tree automata, I describe the models of a CTL formula by an alternating tree automaton, following the ideas of Kupferman, Vardi, and Wolper, [2], but using the more general model of automaton.

Let φ be an arbitrary CTL formula and P the set of propositional variables occurring in φ. The ATA $\boldsymbol{A}_\varphi$ is defined by $\boldsymbol{A}_\varphi = (Q, P, \varphi, \delta, \Omega)$ where Q is the set of all CTL subformulas of φ including φ itself, Ω is the Muller acceptance condition that contains all sets of subformulas of φ that do not contain formulas

starting with EU or AU, and δ is defined by induction, where, for instance, the inductive step for EU is given by

$$\delta(\mathsf{EU}(\psi,\chi),a) = \chi \vee (\psi \wedge \Diamond\mathsf{EU}(\psi,\chi)) \ . \tag{4}$$

The other cases are similar and follow the ideas of [2]. Note that on the right-hand side of (4) the boolean connectives $\vee$ and $\wedge$ are part of the syntax of transition conditions.

Similar to [2], one can prove by a straightforward induction on the structure of φ:

Theorem 2. *Let φ be an arbitrary CTL formula of length l. Then $\boldsymbol{A}_\varphi$ is an ATA with at most l states such that $Mod(\varphi) = \mathcal{K}(\boldsymbol{A}_\varphi)$.*

It is quite easy to see that for every ATA there exists an equivalent ATA in normal form with the same number of states. So in order to prove Theorem 1 we only need to show:

Theorem 3. *Every ATA in normal form recognizing $Mod(\varphi_n)$ has at least $\binom{n}{\lceil n/2\rceil}$ states.*

5 The General Setting

The method I use to prove Theorem 3 (a cut-and-paste argument) does not only apply to the specific properties defined by the φ_n's but to a large class of "path properties." As with many other situations, the method is best understood when presented in its full generality. In this section, I explain the general setting and present the extended version of Theorem 3, namely Theorem 4.

In the following, *word* stands for nonempty string or ω-word. The set of all words over a given alphabet A is denoted by A^∞. A *language* is a subset of $(2^P)^\infty$ where P is some finite subset of Prop. Given a language L over some alphabet 2^P, $\mathsf{E}L$ denotes the class of pointed Kripke structures $(\boldsymbol{K}, w)$ where there exists a maximal path through $(\boldsymbol{K}, w)$ whose labeling (restricted to the propositional variables in P) is an element of L. (Remember that a path through a pointed Kripke structure always starts in its distinguished world.)

Observe that for every n, we clearly have $\mathrm{Mod}(\varphi_n) = \mathsf{E}L_n$ where

$$L_n = \{a_0a_1\cdots \in (2^{P_n})^\infty \mid \forall i(i < n \to \exists j(p_i \in a_j))\}$$

and $P_n = \{p_0,\ldots,p_{n-1}\}$.

Let L be a regular language. We say a family $\{(u_i,u_i')\}_{i<m}$ is a *discriminating family* for L if $u_iu_i' \in L$ and $u_iu_j' \notin L$ for all $i < m$ and all $j < m$ with $j \neq i$. Obviously, the number of classes of the Nerode congruence[2] associated with L is an upper bound for m. The maximum number such that there exists a discriminating family of that size for L is denoted $\iota(L)$.

The generalized version of Theorem 3 now reads:

[2] The Nerode congruence of a language L is the congruence that considers strings u and v equivalent if for every word x (including the empty word), $ux \in L$ iff $vx \in L$.

Theorem 4. *Let L be a regular language. Then every ATA recognizing $\mathsf{E}L$ has at least $\iota(L)$ states.*

Before we turn to the proof of this theorem in the next section, let's apply it to the languages L_n (as defined above) to obtain the desired lower bounds.

Fix an arbitrary positive natural number $n > 1$ and let $m = \lceil n/2 \rceil$ and $t = \binom{n}{m}$. Write N for the set $\{0, \ldots, n-1\}$ and $^-$ for set-theoretic complementation with respect to N. For every $M \subseteq N$, let $u(M)$ be a string over 2^{P_n} of length $|M|$ such that for every $p_i \in M$, the letter $\{p_i\}$ occurs in $u(M)$. (In other words, $u(M)$ should be a sequence of singletons where for each $i \in M$ the singleton $\{p_i\}$ occurs exactly once and no other singleton occurs.) Let $M_0, \ldots, M_{t-1}$ be an enumeration of all m-subsets of N and let $u_i = u(M_i)$ and $u'_i = u(\bar{M}_i)$. Then $\{(u_i, u'_i)\}_{i<t}$ is a discriminating family for L_n, which means $\iota(L_n) \geq \binom{n}{\lceil n/2 \rceil}$.

This together with Theorem 4 implies Theorem 3 and thus also Theorem 1. (Observe that for $n = 1$ the claims of Theorems 3 and 1 are trivial.)

6 Saturation

In this section, I will introduce the key concepts used in the proof Theorem 4, state the main lemmas, provide as much intuition as is possible within the page limit, and give a rough outline of the proof of Theorem 4.

We will see trees in two different roles. On the one hand, we will look at runs of ATA's, and runs of ATA's are trees by definition. On the other hand, we will consider Kripke structures that are trees. In order to not get confused, I will strictly follow the notational conventions introduced earlier, for instance, that the labeling function of a run $\boldsymbol{R'}$ is referred to by λ'. As we will only work with Kripke structures that are trees, I will use the term *Kripke tree.* A Kripke tree will also be viewed as a pointed Kripke structure where the root of the tree is the distinguished node.

For the rest of this section, fix a language L over some alphabet 2^P, and an ATA $\boldsymbol{A}$. For each state q, write $\boldsymbol{A}_q$ for the ATA that results from $\boldsymbol{A}$ by changing its initial state to q and $\mathcal{K}_q$ for the class $\mathcal{K}(\boldsymbol{A}_q)$, the class of pointed Kripke structures recognized by $\boldsymbol{A}_q$.

Let u be a string. A state q is *preventable* for u if there exists a Kripke tree $\boldsymbol{K}$ such that $u{\cdot}\boldsymbol{K} \notin \mathsf{E}L$ and $\boldsymbol{K} \notin \mathcal{K}_q$. We write $pvt(u)$ for set of all states preventable for u, and for every $q \in pvt(u)$, we pick, once and for all, a Kripke tree $\boldsymbol{K}$ as above and denote it by $\boldsymbol{K}^u_q$. The important observation here is that if $\boldsymbol{K}$ is a Kripke tree, $w \in W$, and $q \in pvt(\boldsymbol{K}{\restriction}w)$, then $\boldsymbol{K'}$ defined by $\boldsymbol{K'} = \boldsymbol{K} \cdot (w, \boldsymbol{K}^u_q)$ with $u = \boldsymbol{K}{\restriction}w$ has the following two properties. First, if $\boldsymbol{K} \notin \mathsf{E}L$, then $\boldsymbol{K'} \notin \mathsf{E}L$. Second, there is no run $\boldsymbol{R}$ of $\boldsymbol{A}$ on $\boldsymbol{K'}$ that has a node v with $w_{\boldsymbol{R}}(v) = w$ and $\boldsymbol{K'}, \boldsymbol{R}, v \models \Box q$. In a certain sense, by adding $\boldsymbol{K}^q_u$ to $\boldsymbol{K}$, the condition $\Box q$ is "prevented" from being used at w.

A state q is *always successful* for u if there exists a state $q' \in pvt(u)$ such that $\boldsymbol{K}^u_{q'} \in \mathcal{K}_q$. We write $scf(u)$ for the set of states always successful for u, and for every $q \in scf(u)$, we pick, once and for all, a state q' as above and denote it

by q^u. (Note that whether or not a state is always successful for a string depends on the particular choices for the $\boldsymbol{K}_q^u$'s.) The important observation here is the following. Choose $\boldsymbol{K}$, w, u, and $\boldsymbol{K'}$ as in the previous paragraph. If $q \in \mathit{scf}(u)$ and if we want to construct a run $\boldsymbol{R}$ of $\boldsymbol{A}$ on $\boldsymbol{K'}$, then we can always make sure that $\boldsymbol{K}, \boldsymbol{R}, v \models \Diamond q$ holds for a node v with $w_{\boldsymbol{R}}(v) = w$, because we only need to add to $\boldsymbol{R}$ a successful run of $\boldsymbol{A}$ on $\boldsymbol{K}_{q'}^u$ with $q' = q^u$. Formally, if $\boldsymbol{R}_{q'}^u$ is such a run, we only need to consider $\boldsymbol{R} \cdot (v, \boldsymbol{R}_{q'}^u)$ instead of $\boldsymbol{R}$.

A world w of a Kripke tree $\boldsymbol{K}$ is said to be *saturated* if for every $q \in \mathit{pvt}(\boldsymbol{K}{\uparrow}w)$, there exists $w' \in \mathit{Scs}(\boldsymbol{K}, w)$ such that $\boldsymbol{K}{\downarrow}w'$ is isomorphic to $\boldsymbol{K}_q^u$ with $u = \boldsymbol{K}{\uparrow}w$.

Let $\boldsymbol{K}$ be an arbitrary Kripke tree. The Kripke tree $\boldsymbol{K}^{\mathbf{s}}$ is defined by

$$\boldsymbol{K}^{\mathbf{s}} = \boldsymbol{K} \cdot \{(w, \boldsymbol{K}_q^u) \mid w \in \mathit{In}(\boldsymbol{K}),\ u = \boldsymbol{K}{\uparrow}w, \text{ and } q \in \mathit{pvt}(u)\} \ , \tag{5}$$

that is, in $\boldsymbol{K}^{\mathbf{s}}$, every inner world from $\boldsymbol{K}$ is saturated.

Remark 1. Let $\boldsymbol{K}$ be an arbitrary Kripke tree. If $\boldsymbol{K} \in \mathsf{E}L$, then $\boldsymbol{K}^{\mathbf{s}} \in \mathsf{E}L$.

This is because every maximal path through $\boldsymbol{K}$ is also present in $\boldsymbol{K}^{\mathbf{s}}$; no successors are added to leaves.

Let τ be an arbitrary transition condition over Q and $X, Y \subseteq Q$. The *X-Y-reduct* of τ, denoted $\tau^{X,Y}$, is obtained from τ by replacing

— every atomic subformula of the form $\Box q$ with $q \in X$ by 0,
— every atomic subformula of the form $\Box q$ with $q \in Q \setminus X$ by 1, and
— every atomic subformula of the form $\Diamond q$ with $q \in Y$ by 1.

Let $\boldsymbol{K}$ be an arbitrary Kripke tree. A *partial run* of $\boldsymbol{A}$ on $\boldsymbol{K}$ is defined just as an ordinary accepting run with the following modification of local consistency as defined in (2). For every $v \in V$ such that $w_{\boldsymbol{R}}(v) \in \mathit{In}(\boldsymbol{K})$, it is required that

$$\boldsymbol{K}, \boldsymbol{R}, v \models \tau_v^{X_v, Y_v} \tag{6}$$

holds where τ_v is as defined in (3) and

$$X_v = \mathit{pvt}(\boldsymbol{K}{\uparrow}w_{\boldsymbol{R}}(v)) \ , \qquad Y_v = \mathit{scf}(\boldsymbol{K}{\uparrow}w_{\boldsymbol{R}}(v)) \ .$$

Note that in general neither τ implies $\tau^{X,Y}$ nor $\tau^{X,Y}$ implies τ. So there is no a priori relation between the existence of runs and partial runs. But using Remark 1 and the right notion of restriction of a run one can prove the following.

Lemma 1. *Let $\boldsymbol{K}$ be an arbitrary Kripke tree. Assume $\mathcal{K}(\boldsymbol{A}) = \mathsf{E}L$ and $\boldsymbol{K} \in \mathcal{K}(\boldsymbol{A})$. Then there exists a partial run of $\boldsymbol{A}$ on $\boldsymbol{K}$.*

Let $\boldsymbol{R}$ be a partial run of $\boldsymbol{A}$ on a Kripke tree $\boldsymbol{K}$. The run $\boldsymbol{R}$ is *distributed* if for every $w \in W$ there exists at most one $v \in V$ with $w_{\boldsymbol{R}}(v) = w$.

The set of all *frontier pairs* of $\boldsymbol{R}$, denoted by $\mathit{FrtPrs}(\boldsymbol{R})$, is defined by $\mathit{FrtPrs}(\boldsymbol{R}) = \{\lambda(v) \mid v \in V \text{ and } w_{\boldsymbol{R}}(v) \in \mathit{Lvs}(\boldsymbol{K})\}$. Similarly, the set of all *frontier states* of $\boldsymbol{R}$, denoted $\mathit{FrtSts}(\boldsymbol{R})$, is defined by $\mathit{FrtSts}(\boldsymbol{R}) = \{q_{\boldsymbol{R}}(v) \mid v \in V \text{ and } w_{\boldsymbol{R}}(v) \in \mathit{Lvs}(\boldsymbol{K})\}$.

The crucial lemma connecting Kripke trees with saturated inner worlds and partial runs is as follows.

Lemma 2. *Let $\boldsymbol{K}$ be a Kripke tree and $\boldsymbol{R}$ a distributed partial run of $\boldsymbol{A}$ on $\boldsymbol{K}$. Assume that for every $q \in \mathit{FrtSts}(\boldsymbol{R})$ there exists a Kripke tree $\boldsymbol{K}_q \in \mathcal{K}_q$ such that the tree $\boldsymbol{K}^*$ defined by*

$$\boldsymbol{K}^* = \boldsymbol{K} \odot \{(w, \boldsymbol{K}_q) \mid q \in \mathit{FrtSts}(\boldsymbol{R})\}$$

does not belong to EL.

Then there exists an accepting run of $\boldsymbol{A}$ on the Kripke tree $\boldsymbol{K}^{**}$ defined by

$$\boldsymbol{K}^{**} = \boldsymbol{K}^s \odot \{(w, \boldsymbol{K}_q) \mid (w, q) \in \mathit{FrtPrs}(\boldsymbol{R})\} \ ,$$

which does not belong to EL.

Note that because $\boldsymbol{R}$ is supposed to be distributed, the trees $\boldsymbol{K}^*$ and $\boldsymbol{K}^{**}$ are obtained from $\boldsymbol{K}$ and $\boldsymbol{K}^{\mathbf{s}}$, respectively, by adding to each leaf at most one of the trees $\boldsymbol{K}_q$.

The proof of this lemma is technically involved and makes extensive use of the aforementioned properties of preventable and always successful states.

I will conclude this section with a rough sketch of the proof of Theorem 4.

Sketch of the Proof of Theorem 4. Let $\{(u_i, u_i')\}_{i<m}$ be a discriminating family for L of size $\iota(L)$ and $\boldsymbol{A}$ an ATA with $\mathcal{K}(\boldsymbol{A}) = \mathsf{E}L$. I claim that for every $i < m$, there exists a state q such that $u_i' \in \mathcal{K}_q$, but $u_j' \notin \mathcal{K}_q$ for $j < m$ and $j \neq i$. This clearly implies the claim of the theorem.

By way of contradiction, assume this is not the case. Then there exists $i < m$ such that for every $q \in Q$ with $u_i' \in \mathcal{K}_q$ there exists $j \neq i$ such that $u_j' \in \mathcal{K}_q$. For every such q let j_q be an appropriate index j.

Let $\boldsymbol{K}$ be a $|Q|$-branching Kripke tree[3] such that every maximal path starting with the root is labeled $u_i a_i$ where a_i is the first letter of u_i'. Consider the Kripke tree $\boldsymbol{K'}$ defined by $\boldsymbol{K'} = \boldsymbol{K} \odot \{(w, u_i') \mid w \in \mathit{Lvs}(\boldsymbol{K})\}$.

Clearly, $\boldsymbol{K'} \in \mathsf{E}L$ (because every maximal path through $\boldsymbol{K'}$ is labeled $u_i u_i'$). Thus, by Lemma 1, there exists a partial run of $\boldsymbol{A}$ on $\boldsymbol{K'}$. By restricting this run to the worlds in $\boldsymbol{K}$, we obtain a partial run of $\boldsymbol{A}$ on $\boldsymbol{K}$. This run has the obvious property that for every $q \in \mathit{FrtSts}(\boldsymbol{R})$ there exists an accepting run of $\boldsymbol{A}_q$ on u_{j_q}'. By manipulating this run adequately, using the fact that $\boldsymbol{K}$ is $|Q|$-branching, one can transform it into a distributed partial run with the same property. This run together with the u_{j_q}''s replacing the $\boldsymbol{K}_q$'s satisfies the assumptions of Lemma 2. We can thus conclude the Kripke tree $\boldsymbol{K}^{**}$ as defined in Lemma 2, which does not belong to $\mathsf{E}L$, is accepted by $\boldsymbol{A}$—a contradiction.

7 Connection with Büchi Automata and μ-Calculus

One can show that Theorem 2 also holds for the modal μ-calculus (see, for instance, [2]). So we also obtain: every modal μ-calculus formula equivalent to

[3] A Kripke tree $\boldsymbol{K}$ is *m-branching* if for every world $w \in W$ the following is true. For every successor w_0 of w there exist at least $m-1$ other successors $w_1, \ldots, w_{m-1}$ of w such that all subtrees $\boldsymbol{K}{\downarrow}w_0, \ldots, \boldsymbol{K}{\downarrow}w_{m-1}$ are isomorphic.

the CTL^+ formula φ_n has length at least $\binom{n}{\lceil n/2 \rceil}$. This is interesting because of the following.

As the modal μ-calculus is closed under syntactic negation, the above also says that every modal μ-calculus formula equivalent to the CTL^+ formula

$$\mathsf{A}(\mathsf{G}\neg p_0 \vee \cdots \vee \mathsf{G}\neg p_{n-1})$$

has length at least $\binom{n}{\lceil n/2 \rceil}$. And, clearly, this property can easily be expressed by an alternation-free μ-calculus (AFMC) formula (according to the definition of alternation-freeness as introduced by Emerson and Lei in [7]), because it can be expressed in CTL. On the other hand, the set of all ω-words over 2^{P_n} satisfying the linear-time temporal property $\mathsf{G}\neg p_0 \vee \cdots \vee \mathsf{G}\neg p_{n-1}$ is recognized by a nondeterministic Büchi word automaton (NBW) with $n + 1$ states. We therefore have:

Corollary 1. *There is an exponential lower bound for the translation NBW $\mapsto$ AFMC in the sense of [10].*

This answers a question left open by Kupferman and Vardi in [10].

8 Conclusion

We have seen that there is an exponential gap between the succinctness of CTL^+ and CTL, as well as an exponential gap between nondeterministic Büchi word automata and alternation-free μ-calculus. Just as in many other situations, the automata-theoretic approach to understanding the expressive power of (specification) logics has proved to be useful.

References

1. O. Bernholtz [Kupferman] and O. Grumberg. Branching temporal logic and amorphous tree automata. In E. Best, ed., *CONCUR'93*, vol. 715 of *LNCS*, 262–277. 111
2. O. Bernholtz [Kupferman], M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In D. L. Dill, ed., *CAV '94*, vol. 818 of *LNCS*, 142–155. 111, 115, 116, 116, 119
3. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications: A practical approach. In *PoPL '83*, 117–126. 111
4. E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *STOC '82*, 169–181. 110, 111, 111
5. E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. System Sci.*, 30(1):1–24, 1985. 110, 111, 111, 113
6. E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In C. Courcoubetis, ed., *CAV '93*, vol. 697 of *LNCS*, 385–396. 111
7. E. A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In *LICS '86*, 267–278. 120

8. K. Etessami, M. Y. Vardi, and Th. Wilke. First-order logic with two variables and unary temporal logic. In *LICS '97*, 228–235. 111, 111
9. J. A. W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, Calif., 1968. 110
10. O. Kupferman and M. Y. Vardi. Freedom, weakness, and determinism: From linear-time to branching-time. In *LICS '98*, 81–92. 111, 111, 120, 120
11. L. J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Dept. of Electrical Engineering, MIT, Boston, Mass., 1974. 110
12. Th. Wilke. *CTL$^+$ is exponentially more succinct than CTL*. Technical Report 99-7, RWTH Aachen, Fachgruppe Informatik, 1999. Available online via ftp://ftp.informatik.rwth-aachen.de/pub/reports/1999/index.html. 112