The Common Fragment of ACTL and LTL

Mikołaj Bojańczyk *

Warsaw University

Abstract. The paper explores the relationship between tree languages definable in LTL, CTL, and ACTL, the fragment of CTL where only universal path quantification is allowed. The common fragment of LTL and ACTL is shown to be strictly smaller than the common fragment of LTL and CTL. Furthermore, an algorithm is presented for deciding if an LTL formula can be expressed in ACTL. This algorithm uses an effective characterization of level 3/2 of the concatenation hierarchy for infinite words, also a new result.

Two of the most commonly used logics in verification are LTL and CTL. The first is a linear time logic, a formula of LTL describes a property of words. To describe properties of trees, one applies universal path quantification: an LTL formula is valid in a tree if it is valid in all paths. CTL, on the other hand, is a branching time logic. A formula of CTL refers explicitly to the branching in the tree, by using both universal and existential path quantification.

What is the relationship between the two logics? Which LTL definable properties can be defined in CTL, and which CTL definable properties can be defined in LTL? In other words, what is the common fragment of CTL and LTL?

There is a well known algorithm, which given an automaton on infinite trees, determines if its language can be defined in LTL (basically, a tree constructed by mixing paths of different trees accepted by the automaton, must still be accepted by the automaton; furthermore, the appropriate word language must be aperiodic). For tree languages defined in CTL* there is also a simple characterization of Clarke and Draghilescu: a CTL* formula is equivalent to an LTL formula, if and only if it is equivalent to the one obtained by removing the path quantifiers [3]. Maidl [6] has shown that if the input is given as a CTL formula, then problem of LTL definability becomes PSPACE complete.

The converse question, however, remains an open problem: is it decidable if a given LTL formula can be equivalently written as a CTL formula? A second, more general, problem is to decide if an arbitrary regular language of infinite trees can be defined in CTL. It seems a good idea to begin with the first problem before tackling the second one.

If an LTL formula with universal path semantics can be defined by a CTL formula, then why should the CTL formula use existential modalities, such as "exists a successor with φ"? Shouldn't it be enough to consider ACTL formulas, where only universal path quantification is used? The first result of this paper is

* Author supported by Polish government grant no. N206 008 32/0810.

that, possibly surprisingly, this assumption is wrong. Indeed, a very simple LTL property, "all paths belong to $(ab)^*a(ab)^*c^\omega$", can be defined in CTL but not ACTL. Intuitively speaking, to catch the extra a on every path, existential path quantification is needed.

Therefore, two distinct questions can be investigated: which LTL properties can be defined in CTL, and which LTL properties can be defined in ACTL. The other main result of this paper is an effective characterization of the second common fragment: one can decide if an LTL formula φ can be expressed in ACTL. This problem has already been considered in [6], where it was shown that a necessary and sufficient condition for ACTL definability is that $\neg\varphi$, when seen as a word language, can be recognized by a certain restricted type of Büchi automaton. This condition, however, was not known to effective, i.e. there was no algorithm that decided if $\neg\varphi$ could be recognized by the restricted Büchi automaton.

The second contribution of this paper is such an algorithm. It is easy to see that the restricted Büchi automata defined in [6] are equivalent to ω-regular languages on level 3/2 of the concatenation hierarchy, i.e. finite unions of expressions

$$A_0^* a_1 A_1^* a_2 \cdots A_{k-1}^* a_k A_k^\omega \ .$$

Therefore, deciding if an LTL formula can be defined in ACTL boils down to testing if an ω-regular language belongs to level 3/2 of the concatenation hierarchy. This problem was known to be decidable for finite words [1, 2, 8]. We generalize this result to infinite words. In the process, we also present a simplified proof for finite words.

The paper is organized as follows. In Section 1, we show that the common fragment of LTL and CTL is strictly larger than the common fragment of LTL and ACTL. Section 2 gives an effective characterization of those LTL properties that can be defined in ACTL. Finally, in Section 4, we present concluding remarks. These concern mainly the common fragment of LTL and CTL, about which little is known.

1 The common fragment of CTL and LTL needs existential modalities

Trees in this paper are unordered, infinite and unranked. In other words, a tree is a connected directed graph with nodes of indegree at most one, but outdegree at least one. The last condition is so that every (maximal) path in the tree is infinite. Trees are labeled.

The results in this paper would also apply to finite trees, or transition systems. Some of the results would be cleaner for finite trees, we will come back to this at the end of the paper.

LTL is a linear time temporal logic. An LTL formula specifies a property of an infinite word. (When we say a word position satisfies φ, we mean that the

suffix beginning in that position satisfies φ.) The modalities are: $\varphi \mathsf{U} \psi$ (there is a position with ψ, and all preceding positions satisfy φ), $\mathsf{X}\varphi$ (the second word position satisfies φ) and $\mathsf{G}\varphi$ (all positions in the word satisfy φ). Furthermore, boolean connectives and label tests (the formula a describes words that begin with a) are allowed. Kamp's theorem [5] says that LTL has the same expressive power as first-order logic with the linear order on word positions. An LTL formula can be evaluated in a tree, it is said to be valid if all maximal paths in the tree satisfy it. In this sense, very simple tree properties, such as "some node in the tree has label a", cannot be defined in LTL. In the following, we will indicate whether an LTL formula is understood to define a word language, or a tree language.

CTL [4] is a temporal branching time logic, i.e. its modalities explicitly quantify over tree paths, possibly existentially. A CTL formula specifies a property of a tree. As with LTL, when we say a formula holds in a tree node (or equivalently, on a position on a path in the tree), we mean that the subtree of that node satisfies the formula. The modalities are: $\mathsf{A}\varphi \mathsf{U} \psi$ (on every path, there is a position with ψ, and all preceding positions on the path satisfy φ), $\mathsf{AX}\varphi$ (every successor of the root satisfies φ) and $\mathsf{AG}\varphi$ (on every path, every position satisfies φ). Furthermore, boolean connectives and label tests (the formula a describes trees whose root has label a) are allowed. Existential quantification can be simulated using negation. In other words, CTL is obtained from LTL by adding universal path quantification next to every modality. In particular, over trees with only one path, i.e. where all nodes have exactly one successor, CTL has the same expressive power as LTL. In general, however, the two logics diverge, for instance CTL cannot express $\mathsf{FG}a$ (on every path, finitely many non a labels).

ACTL is the fragment of CTL that does not allow negation, i.e. where only universal path quantification is allowed. (Here, the atomic propositions are node labels, so they are mutually exclusive; if they are are not exclusive then negation is allowed next to atomic propositions.) Clearly, ACTL is a proper fragment of CTL; for instance, the CTL property "some node has label a" is not definable in ACTL.

The main result in this section is:

Theorem 1. *The language L = "all paths belong to $(ab)^*a(ab)^*c^\omega$" is definable in CTL and LTL, but not ACTL.*

This result is somewhat surprising: the language L talks about all paths, while the corresponding CTL formula must quantify existentially over paths. This example shows that ideas significantly different from those in [6] are needed to understand the common fragment of LTL and CTL.

Before proceeding with the proof, we would like to remark the similarity of this "paradox" with a result for first-order logic over finite binary trees. In [9], Potthof showed that the language "all paths belong to $(aa)^*$" is definable in first-order logic over finite binary trees, even though the word language $(aa)^*$ is not definable in first-order logic over words. His technique was similar to the one invoked below, in that it used properties of "maximal" nodes.

We will now prove Theorem 1.

Lemma 1. *The language L is definable in CTL.*

Proof
It is easy to show that the language "all paths belong to $(ab)^*c^\omega$" is definable in CTL. Let φ_a be such a formula; we will use it below. Likewise we will use a formula φ_b for the language "all paths belong to $b(ab)^*c^\omega$".

The formula for the language L is a conjunction of several properties. First, we have to manage the way c's are used. Formula (1) says that every path contains some c, and every time c appears, all subsequent nodes are c's; finally, only b nodes can have a c successor:

$$\mathsf{AF}c \ \wedge \ \mathsf{AG}(c \Rightarrow \mathsf{AG}c) \ \wedge \ \mathsf{AG}(a \Rightarrow \mathsf{AX}(a \vee b)) \ . \tag{1}$$

Formula (2) says that the tree does not contain two consecutive b's:

$$\mathsf{AG}(b \Rightarrow \mathsf{AX}(a \vee c)) \ . \tag{2}$$

Formula (3) says that on every path, two consecutive a's can be found at most once:

$$\neg\mathsf{EF}(a \wedge \mathsf{EX}(a \wedge (\mathsf{EF}(a \wedge \mathsf{EX}a))) \ . \tag{3}$$

So far, we have stayed within ACTL. The above three properties guarantee that every path in the tree is either in $(ab)^*c^\omega$ or in $(ab)^*a(ab)^*c^\omega$, as long as the root has label a. We now need to eliminate the paths of the first type. First, we enforce the root label, and say that at least one path is not in $(ab)^*c^\omega$

$$a \quad \wedge \quad \neg\varphi_a \ . \tag{4}$$

Note that already here, we go beyond ACTL, since φ_a is negated. The more important property, however, says there is no node x such that: some path beginning in x has two consecutive a's, and some successor of x satisfies φ_a or φ_b, depending on the label of x. This expressed by the formula:

$$\neg\mathsf{EF}(\mathsf{EF}(a \wedge \mathsf{EX}a) \wedge (a \Rightarrow \mathsf{EX}\varphi_b) \wedge (b \Rightarrow \mathsf{EX}\varphi_a)) \tag{5}$$

Here again we go beyond ACTL. We claim that a tree belongs to L if and only if it satisfies the conjunction of formulas (1)-(5).

The left-to-right implication is proved as follows. Let t be a tree in L. Clearly (1)-(4) have to be satisfied. For (5), we need to show that every node x fails the property:

$$\mathsf{EF}(a \wedge \mathsf{EX}a) \wedge (a \Rightarrow \mathsf{EX}\varphi_b) \wedge (b \Rightarrow \mathsf{EX}\varphi_a) \ .$$

We only consider the case when the node x has label a, the other is done in a similar way. Let then x be a node with label a that satisfies $\mathsf{EF}(a \wedge \mathsf{EX}a)$. By (1)-(4), the path leading up to x must belong to $(ab)^*$. In particular, no successor of x can be the beginning of a path in $(ab)^*c^\omega$, not to mention having all paths of this form (which is what $\mathsf{EX}\varphi_a$ says).

We now take the right-to-left implication. Let then t be a tree that satisfies formulas (1)-(4). We claim that if t is outside L, then the formula (5) fails. By (1)-(4), the tree has paths of the form $(ab)^*c^\omega$, and of the form $(ab)^*a(ab)^*c^\omega$. Let X be the set of nodes, which lie on the intersection of some path of the form $(ab)^*a(ab)^*c^\omega$ and some other path of the form $(ab)^*c^\omega$. By $\mathsf{AG}c$, the prefix-closed set X does not contain an infinite path, therefore it has some maximal element, i.e. a node $x \in X$ without proper descendants in X. (Since nodes may have infinite outdegree, there may be no common bound on the depth of nodes from X, but this is not a problem here, since we only need one maximal node.) We claim that the maximal node x witnesses the failure of (5), i. e. x satisfies

$$\mathsf{EF}(a \wedge \mathsf{EX}a) \wedge (a \Rightarrow \mathsf{EX}\varphi_b) \wedge (b \Rightarrow \mathsf{EX}\varphi_a) \ .$$

We only consider the case where x has label b. By maximality of x, there must be a successor y such that all paths that pass through y belong to $(ab)^*c^\omega$; otherwise y would belong to X. The node y witnesses $\mathsf{EX}\varphi_a$ (note that y may have label c, and only c's in its subtree). Since the path leading to x belongs to $(ab)^*$ and x is on some path in $(ab)^*a(ab)^*c^\omega$, there must be two consecutive a's on some path that begins in x. □

We now show that the language L is not definable in ACTL. We will use a characterization of Maidl from [6], slightly restated:

Theorem 2. *Let $L \subseteq A^\omega$ be a language of infinite words. The following are equivalent:*

- *The tree language "all paths belong to L" is definable in ACTL;*
- *The complement of L is a finite union of languages of the form*

$$\begin{array}{c} A_0^* a_1 A_1^* a_2 \cdots A_{n-1}^* a_n A_n^\omega \\ a_1, \ldots, a_n \in A,\ A_1, \ldots, A_{n+1} \subseteq A \ . \end{array} \tag{6}$$

Languages that are finite unions of expressions as in (6) are also known as languages on *level 3/2 of the concatenation hierarchy*, see [7] for a more thorough description of this and other levels. Therefore, the second condition in the above theorem is the same as saying the complement of L belongs to level 3/2.

The original statement in [6] was not in terms of level 3/2, but in terms of 1-weak Büchi automata. A 1-weak Büchi automaton is a Büchi automaton where the states are ordered, and a transition can only go down in the order, or stay in the same state. A level 3/2 expression can easily be translated into a 1-weak Büchi automaton, by using nondeterminism of the automata. The translation in the other direction is no more difficult: the subexpressions A_i^* correspond to staying in the same state for some time; while A_n^ω corresponds to an infinite self-loop in an accepting state. The finite union corresponds to the finitely many possible paths in the order.

Lemma 2. *The language L is not definable in ACTL.*

Proof
Towards a contradiction with Theorem 2, assume that the complement of L is defined by an finite union of expressions as in (6). Let n be the size of the largest of these expressions. Since $(ab)^{n+1}c^\omega$ is outside L, it must be captured by one of the expressions:

$$(ab)^{n+1}c^\omega \in A_0^* a_1 A_1^* a_2 \cdots A_{m-1}^* a_m A_m^\omega \ .$$

Since $m \leq n$, it is fairly easy to see that the word $(ab)^j a (ab)^{n+1-j} c^\omega \in L$ will also be captured by the same expression, a contradiction. □

In the next section we give an algorithm that decides if a word language can be defined on level 3/2; therefore the above proof could be replaced by just running the algorithm on L (and reading the output "no").

We remark that a third equivalent condition can be added to Theorem 2: the word language L is defined by a Π_2 formula, i.e. a first order formula with a quantifier prefix $\forall^*\exists^*$, where the signature allows label tests, and the linear order on word positions.

2 Effective characterization of level 3/2 for infinite words

In this section give show that the following problem is decidable:

Input: a regular word language $L \subseteq A^*$ (resp. $L \subseteq A^\omega$).
Question: is L on level 3/2 of the concatenation hierarchy?

The case when L is a language of finite words—when $L \subseteq A^*$—is treated in Section 2.2, while the infinite case—when $L \subseteq A^\omega$—is treated in Section 2.3.

The result on finite words is not new. The first proof is due to Arfi [1, 2] and uses a difficult result of Hashiguchi. A different proof was presented by Pin and Weil in [8], one of the advantages being a better complexity for the algorithm. Instead of just citing these results, we present a complete proof below. There are two reasons. The first reason is that although the proof in [8] is self-contained, it does use a number of involved and general algebraic concepts. The proof below is specifically tailored to level 3/2, although the underlying ideas are similar to [8], in particular the use of Simon's factorization forests. The second reason is that we are actually interested in infinite words, for which the result has not yet been shown. Although the infinite case turns out to be a straightforward adaptation of the finite one, its proof would be difficult to understand without the finite case.

The proof will use an algebraic language, especially monoids and morphisms. Recall that a monoid is a set S together with an associative concatenation operation, denoted multiplicatively. Furthermore, there is an identity element. An example of a monoid is the *free monoid over* A, i.e. the set A^* of all words over the alphabet A (the identity element is the empty word). Finite monoids are used to recognize regular languages, just as automata. In order to recognize a regular language $L \subseteq A^*$, we use a morphism $\alpha : A^* \to S$. (A *morphism* is a function

that preserves concatenation, and the identity element). A morphism is said to recognize L, if membership $w \in L$ depends only on the value $\alpha(w) \in S$. In other words, $L = \alpha^{-1}(\alpha(L))$. Languages recognized by morphisms into finite monoids are exactly the same ones as those recognized by finite automata, so morphisms and monoids can be used as a different way of describing regular languages. For each regular language, there is a *syntactic morphism*, whose target monoid is the smallest monoid that can be used to recognize the language. This morphism corresponds to the two-sided Myhill-Nerode equivalence of the language.

An important concept will be the non-connected subword relation. If S is a monoid, and $s, t \in S$, we write $s \preceq t$ if for some $s_1, \ldots, s_n, t_0, \ldots, t_n \in S$ we have

$$s = s_1 \cdots s_n \qquad t = t_0 s_1 t_1 \cdots t_{n-1} s_n s_n \ .$$

This relation is transitive in the free monoid, but need not be transitive in general.

When characterizing level 3/2, we will use a result of Simon about "factorization forests". We present this result in a slightly different, but equivalent, way than the original paper [10]. As mentioned previously, the Simon result was already used in [8]; in this sense our approach to characterizing level 3/2 is not new. The Simon result is presented in the next section, while Sections 2.2 and 2.3 apply it to describing level 3/2.

2.1 Typed regular expressions

In this section, we present regular expressions that are well typed for a morphism. Before looking at the formal definition, consider first the language "even number of $a's$", over an alphabet $\{a, b\}$. This language is described by the regular expression

$$\big((b^*a)(b^*a)b^*\big)^* \ .$$

Consider now a morphism $\alpha : \{a, b\}^* \to \{0, 1\}$, which recognizes the language by counting the number of a's modulo two. It so happens that the regular expression presented above is well-typed for this morphism, i.e. for every subexpression we can indicate the appropriate value assigned by α:

$$b : 0 \quad b^* : 0 \quad a : 1 \quad b^*a : 1 \quad (b^*a)(b^*a) : 0 \quad \big((b^*a)(b^*a)b^*\big)^* : 0$$

Not every regular expression is well typed with respect to every morphism. However, a consequence of the factorization forest theorem of Simon says that every regular language recognized by a morphism α can be defined by a regular expression that is well typed with respect to α. The rest of this section presents this result in more detail.

To simplify notation, we do not distinguish between regular expressions and the languages they describe. In particular, we will denote expressions using the same letters as languages, i.e. L, K and M. Fix a morphism $\alpha : A^* \to S$. An α-typed regular expression L is like a regular expression, but each subexpression

must be typed by a value in the monoid S. The Kleene star is only allowed in the form L^+ with nonempty iterations, and furthermore L^+ is only allowed if L is typed by an idempotent (an element s with $ss = s$). The formal inductive definition follows:

- Any single word $w \in A^*$ is an α-typed expression typed by $\alpha(s)$.
- If L, K are α-typed expressions both typed by $s \in S$, then $L \cup K$ is an α-typed expression also typed by s.
- If L, K are α-typed expressions typed by $s, t \in S$ respectively, then LK is an α-typed expression typed by st.
- If L is an α-typed expression typed by $s \in S$, and s is idempotent, then L^+ is an α-typed expression also typed by s.

The following result is a straightforward consequence of Simon's factorization forests theorem [10]:

Theorem 3. *Let $\alpha : A^* \to S$ be a morphism. For any $s \in S$, the language $\alpha^{-1}(s)$ can be defined by an α-typed expression. In particular, any language recognized by α can be defined by a finite union of α-typed expressions.*

2.2 Characterization of level 3/2 for finite words

In this section give show that the following problem is decidable:

Input: a regular word language $L \subseteq A^*$.
Question: is this language on level 3/2 of the concatenation hierarchy?

For the sake of completeness, in the theorem below we also include the equivalence between level 3/2 of the concatenation hierarchy and level Σ_2 of the first-order quantification hierarchy, a special case of a result by Thomas [11]. Recall that a sentence of Σ_2 is a first order sentence with quantifier prefix $\exists^*\forall^*$. A sentence defines the set of words where it holds, in a given word quantification is over word positions. The signature contains the linear order on word positions (but not the successor), and label tests. For instance, the following Σ_2 sentence defines the language $a^*ba^*ca^*$:

$$\exists x_1, x_2 \forall y \quad x_1 < x_2 \wedge b(x_1) \wedge c(x_2) \wedge \big(y \neq x_1 \wedge y \neq x_2 \Rightarrow a(y)\big)$$

The key point in the algorithm will be a type of pumping relation that is complete for level 3/2, i.e. all languages on level 3/2 are closed under this type of pumping and, conversely, all languages closed under this type of pumping are on level 3/2. Consider the following rewriting rule (on words in A^*):

$$w_1 w_2 \quad \to_\alpha \quad w_1 v w_2 \qquad \text{if } \alpha(v) \preceq \alpha(w_1) = \alpha(w_2) = \alpha(w_1 w_2). \tag{7}$$

(Recall that $\preceq$ is the non-connected subword relation.) We call this a rule, but it is more properly called a rule scheme, since there are infinitely many words w_1, w_2, v with the above properties. Let $\Rightarrow_\alpha$ be the rewriting system generated

by this rule, i.e. $w \Rightarrow_\alpha v$ holds if v can be obtained from w by applying the rule $\rightarrow_\alpha$ to infixes a number, possibly zero, of times. We will treat this rewriting system as a pumping relation. We define $cl_\alpha(L)$ to be the set of words w such that $v \Rightarrow_\alpha w$ holds for some $v \in L$.

Theorem 4. *For a regular language L of finite words, the following are equivalent:*

1. *L is definable by a sentence of Σ_2;*
2. *L is on level 3/2 of the concatenation hierarchy, i.e. equivalent to a finite union of expressions $A_0^* a_1 A_1^* \cdots A_{n-1}^* a_n A_n^*$;*
3. *$L = cl_\alpha(L)$ holds for α the syntactic morphism of L.*

We do not yet show that this characterization is effective, this is the subject of Section 3.

The implication from 2 to 1 is immediate, while the implication from 1 to 3 can be shown using a standard Ehrenfeucht-Fraisse argument. We concentrate on the implication from 3 to 2.

Lemma 3. *For every α-typed expression L, there is level 3/2 expression K with $L \subseteq K \subseteq cl_\alpha(L)$.*

Before we proving this lemma, we show how it gives the implication from 3 to 2 in Theorem 4. Let $L \subseteq A^*$ be a language whose syntactic morphism is $\alpha : A^* \to S$, and which satisfies $L = cl_\alpha(L)$. By Theorem 3, L can be defined as a union of α-typed expressions $L_1 \cup \cdots \cup L_n$. Let $K_1, \ldots, K_n$ be the languages obtained by applying the lemma to $L_1, \ldots, L_n$. We claim that the union of the languages K_i is the same as L, which concludes the proof. Indeed,

$$L = \bigcup L_i \subseteq \bigcup K_i \subseteq \bigcup_\alpha cl(L_i) = cl(L) = L \ .$$

Proof
The proof is by induction on the α-expression. The induction base is trivial. Concatenation is a consequence of closure of level 3/2 expressions under concatenation and of

$$cl(L_1) \cdot cl(L_2) \subseteq cl(L_1 \cdot L_2) \ .$$

Union is solved the same way. Only the step L^+ remains. Let $e \in S$ be the idempotent that types the expression L, and let K be the level 3/2 expression obtained by applying the induction assumption to L. We need to find a language M for L^+. We set:

$$M = K \cup KB^*K \qquad \text{where } B = \{b \in A : \alpha(b) \preceq e\}$$

Clearly the expression for M is on level 3/2. It remains to show that M satisfies:

$$L^+ \subseteq M \subseteq cl(L^+) \ .$$

By definition of B, we have $L \subseteq B^*$, and hence the left inclusion holds. Only the right inclusion $M \subseteq cl(L^+)$ remains. Let then w be a word in M. The more difficult case is when $w = w_1w_2w_3$, with $w_1, w_3 \in K$ and $w_2 \in B^*$. By assumption on $K \subseteq cl(L)$, there are words $v_1, v_3 \in L$ with $v_1 \Rightarrow_\alpha w_1$ and $v_3 \Rightarrow_\alpha w_3$. The desired conclusion—actually, a stronger result: $M \subseteq cl_\alpha(LL)$—follows by

$$v_1v_3 \Rightarrow_\alpha v_1w_2v_3 \Rightarrow_\alpha w_1w_2w_3$$

The first rewriting uses $\alpha(w_2) \preceq e$, which follows by definition of w_2 and idempotency of e. The second rewriting follows by the assumption on v_1, v_3. □

2.3 Characterization of Σ_2 for infinite words

In this section we will be working with infinite words (ω-words). Our approach will be the same, in particular we will need to use a syntactic monoid $\alpha : A^* \to S$ for a language of infinite words. What is a syntactic monoid for a language $L \subseteq A^\omega$ of infinite words? It is also obtained by using a Myhill-Nerode congruence. Two finite v, v' words are called *L-equivalent* if both:

- For every $u \in A^*$ and $w \in A^\omega$, either both or none of $uvw, uv'w$ are in L.
- For every $u, w \in A^*$, either both or none of $u(vw)^\omega, u(v'w)^\omega$ are in L.

It turns out that L-equivalence is a congruence on A^*, and the mapping that assigns a word its L-equivalence class is a semigroup morphism, called the *syntactic morphism* of an infinite language. See [7] for more details.

We will follow the same approach as in the previous section, by introducing a rewriting relation. This relation will work on infinite words. It is generated by two types of rule. The first type is the same one as in the previous section, i.e. the rule (7), which is used to rewrite finite infixes of the infinite word. The second rule works on infinite suffixes of the word:

$$w^\omega \to_\alpha wuv^\omega \qquad \text{if } \alpha(u), \alpha(v) \preceq \alpha(w) \text{ and } \alpha(w) \text{ is idempotent.} \tag{8}$$

We denote by $\Rightarrow_\alpha^\omega$ the rewriting system generated by both rules (7) and (8). We write $cl_\alpha^\omega(L)$ for the closure of $L \subseteq A^\omega$ under this rewriting system.

Theorem 5. *For a regular language L of infinite words, the following are equivalent:*

1. *L is definable by a sentence of Σ_2;*
2. *L is on level 3/2 of the concatenation hierarchy, i.e. equivalent to a finite union of expressions $A_0^*a_1A_1^* \cdots A_{n-1}^*a_nA_n^\omega$;*
3. *$L = cl_\alpha^\omega(L)$ holds for α the syntactic morphism of L.*

Proof
As for Theorem 4, we only We only do the proof for the implication from 3 to 2.

Let X be the set of pairs $(s,e) \in S^2$ such that e is an idempotent, and some word of the form uv^ω belongs to L, with α mapping u, v to s, e respectively. We first claim that the following equality holds:

$$L \quad = \quad \bigcup_{(s,e)\in X} \alpha^{-1}(s)\alpha^{-1}(e)A_e^\omega \ , \tag{9}$$

where A_e is the set of letters that may appear in a word mapped to e. For the left to right inclusion, fix some word $w \in L$. Using the (infinite) Ramsey theorem, this word can be decomposed as $w = w_0w_1w_2\cdots$, with all the words $w_1, w_2, \ldots$ being mapped by α to the same idempotent e. If s is the value $\alpha(w_0)$, then we clearly have $(s,e) \in X$. Furthermore, clearly all the letters in $w_2, w_3, \ldots$ belong to A_e, which shows that the word w belongs to the right side of (9).

Consider now the right to left inclusion (9). Let uvw be a word belonging to the right hand side, i.e. with $\alpha(u) = s$, $\alpha(v) = e$ and $w \in A_e^\omega$ for some $(s,e) \in X$. As above, there is a decomposition $w = w_0w_1\cdots$ with all the words $w_1, w_2, \ldots$ being mapped to the same element by α. Since all the words $w_1, w_2, \cdots$ are equivalent, it suffices to show that $uvw_0w_1^\omega$ belongs to L. By assumption $w_i \in A_e^*$, we have $\alpha(w_i) \preceq e$ for $i = 0, 1$. Therefore, we have $L \ni uv^\omega \Rightarrow_\alpha^\omega uvw_0w_1^\omega$, and the right hand side of the rewriting must belong to L.

The rest of the reasoning is as in the case for finite words. For each $(s,e) \in X$, we treat $\alpha^{-1}(s)\alpha^{-1}(e)$ as a language $L_{s,e}$ of finite words. Using Lemma 3, we show that the closure $cl_\alpha(L_{s,e})$ of each such language is defined by a level 3/2 expression. As in the previous section, we get an expression for a language that is between L and $cl_\alpha(L)$, this expression must then describe L itself. □

3 Complexity

In this section, we show that the conditions in Theorems 4 and 5 can be effectively checked. To the author's best knowledge, the result below is the first criterion that can be checked in polynomial time with respect to a finite automaton, and not just a semigroup.

Theorem 6. *Given a deterministic finite automaton over finite words, one can decide in polynomial time if the language it recognizes belongs level 3/2 of the concatenation hierarchy.*

Proof
We assume that all states in the automaton are reachable. However, the automaton need not be minimal. The automaton state assumed after reading the word w when beginning in state p is denoted below by pw. Consider the following property, which can be viewed as a forbidden pattern:

(*) Let p, q be states such that for some words $v \preceq w$, $pw = p$, $qw = q$ and $pv = q$ hold. Every word accepted from p is also accepted from q.

We first claim that (*) is equivalent to condition 3 in Theorem 4. Then, we wil show that (*) can be checked in polynomial time.

We begin with the left to right implication in the claim. Take a language L recognized by an automaton where (*) holds, and let α be the syntactic morphism of L. We need to show that the language is closed under applying the rule $\rightarrow_\alpha$. In other words, we need to show that for any words $w_1, w_2, v \in A^*$ with

$$\alpha(v) \preceq \alpha(w_1) = \alpha(w_2) = \alpha(w_1 w_2)$$

and for any two words $u_1, u_2 \in A^*$, we have:

$$u_1 w_1 w_2 u_2 \in L \quad \Rightarrow \quad u_1 w_1 v w_2 u_2 \in L \ .$$

By assumption, there is some word $w \in A^*$ with $v \preceq w$ and $\alpha(w) = \alpha(w_1)$. Recall that for any function $\delta : Q \to Q$ there is some power $k \in \mathbb{N}$ such that $\delta^k = \delta^{2k}$. Since $\alpha(w_1) = \alpha(w_1)\alpha(w_1)$, we may pick the word w so that its transformation on states is idempotent, i.e. $qw = qww$ holds for all w. By assumption on w, and therefore also ww, being equivalent to w_1, w_2 under the morphism α, it is sufficient to show

$$u_1 (ww)(ww) u_2 \in L \quad \Rightarrow \quad u_1 (ww) v (ww) u_2 \in L \ .$$

Let p be the state assumed by the automaton after reading $u_1 w$, and let q be the state pvw. By assumption on w we have $p = pw$ and $q = qw$. Since we have $vw \preceq ww$ and $q = pvw$, we can use the assumption (*) to obtain that any word accepted from p is also accepted from q. But the result follows, since p (resp. q) is the state assumed by the automaton after reading the word on the left (resp. right) hand side of the implication, wihtout the u_2 suffix.

We now show the right to left implication of the claim. Let then p, q and v, w be as in the assumption of (*). Let v_0 a word after reading which the automaton assumes state p, which exists by assumption on all states being accessible. For some power k, we have $\alpha(w^k) = \alpha(w^k)\alpha(w^k)$. This allows us to use the assumption assumption (7) to obtain

$$v_0 w^k w^k u \in L \quad \Rightarrow \quad v_0 w^k v w^k u \in L \qquad \text{for all } u \in A^* \ .$$

Since p (resp. q) is the state assumed by the automaton after reading the word on the left (resp. right) hand side of the implication, wihtout the u suffix, we obtain the desired conclusion of (*): any word accepted from state p is also accepted from state q.

We now show that condition (*) can be tested in polynomial time. The basic idea is that a standard dynamic algorithm for verifying forbidden patterns can be adapted to use the non-connected subword relation $\preceq$. The polynomial time algorithm runs as follows. In a first step, it calculates the tuples of states

$$X = \{(p, p', q, q', r, r') : p' = pw, q' = qw, r' = rv \text{ holds for some } v \preceq w\} \ .$$

This calculation can be done in polynomial time by a least fix-point algorithm: we begin with the tuples that correspond to word pairs $v \preceq w$ of length at

most 1, and then apply the rule

$$(p, p', q, q', r, r'), (p', p'', q', q'', r', r'') \in X \Rightarrow (p, p'', q, q'', r, r'') \in X$$

until X saturates. Once X has been calculated, we identify the pairs (p, q) such that (p, p, q, q, p, q) belongs to X. For each such pair, we verify that all words accepted from p are also accepted from q. □

The same idea can be used for the infinite case:

Theorem 7. *Given a deterministic parity automaton over infinite words, one can decide in polynomial time if the language it recognizes belongs level 3/2 of the concatenation hierarchy.*

Proof
The proof follows the same lines as for finite words, with the difference that we need to check the second rewriting rule (8). This corresponds to the following pattern:

(**) Let p be a state and w a word with $pw = p$, furthermore assume that the top priority labeling the cycle pw is even. For every words $v, u \preceq w$, the word v^ω is accepted from state pwu.

□

A problem with the above theorem is that the input automaton is deterministic parity, and not nondeterministic Büchi, as often used in verification. We do not know if the algorithm can be adapted to nondeterministic automata, or if the exponential blowup due to determinization is indeed necessary.

The above theorem, when combined with Theorem 2, shows that the common fragment of ACTL and LTL is decidable:

Theorem 8. *It is decidable if a regular tree language belongs to the common fragment of ACTL and LTL.*

Proof
Let K be the tree language, given e.g. by a parity tree automaton. We first find if there is a word language $L \subseteq A^\omega$ such that K is the same as "on all paths L". This language, if it exists, can be easily calculated. If the language does not exist, K is clearly not in the common fragment. Otherwise, we we use Theorem 7 to test if if the complement $A^\omega \setminus L$ is on level 3/2 of the concatenation hierarchy. By Theorem 2, this is equivalent to definability in ACTL. □

4 Concluding remarks on CTL

Probably the most interesting remaining problem is this: what are the tree languages that can be defined in CTL? Is there an effective characterization? The most commonly cited tree property that cannot be defined in CTL is "on some path, there are infinitely many a's". However, the weakness of CTL appears not

only in infinitary behavior, and a lot of combinatorially interesting phenomena appear already in finite trees. For instance, the property "the prefix of some path is $(ab)^*c$" cannot be defined in CTL; both over finite and infinite trees (this provides a finitary language that can be expressed in LTL, but not CTL). For finite trees, there are algebraic tools to examine regular languages, such as syntactic objects and morphisms. Even with these tools, it is a nontrivial task to analyze CTL, for instance to show nondefinability of the $(ab)^*c$ language referred to above. But for infinite trees, the situation is exponentially worse, mainly because there is no reasonable notion of a canonical representation of a tree language, or even a deterministic automaton model! Therefore, it seems a good idea to first understand the expressive power of CTL over finite trees, without tackling both finitary and infinitary aspects at the same time.

References

1. M. Arfi. Polynomial operations on rational languages. In *Symposium on Theoretical Aspects of Computer Science*, volume 247 of *Lecture Notes in Computer Science*, pages 198–206, 1987.
2. M. Arfi. Opérations polynomiales et hiérarchies de concaténation. *Theor. Comput. Sci.*, 91(1):71–84, 1991.
3. E. M. Clarke and I. A. Draghicescu. Expressibility results for linear-time and branching-time logics. In *REX Workshop*, pages 428–437, 1988.
4. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logics of Programs: Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.
5. J. A. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Univ. of California, Los Angeles, 1968.
6. M. Maidl. The common fragment of CTL and LTL. In *Foundations of Computer Science*, pages 643–652, 2000.
7. D. Perrin and J.-É. Pin. *Infinite Words*. Elsevier, 2004.
8. J.-É. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory Comput. Systems*, 30:1–30, 1997.
9. A. Potthoff. First-order logic on finite trees. In *Theory and Practice of Software Development*, volume 915 of *Lecture Notes in Computer Science*, pages 125–139, 1995.
10. I. Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72:65 – 94, 1990.
11. W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences*, 25:360–375, 1982.