Theoretical Computer Science 49 (1987) 217-237
North-Holland

THE COMPLEMENTATION PROBLEM FOR BÜCHI AUTOMATA WITH APPLICATIONS TO TEMPORAL LOGIC

A. PRASAD SISTLA
Computer Science Lab, GTE Laboratories Inc, Waltham, MA 02254, U.S.A.

Moshe Y. VARDI
Department K55/802, IBM Almaden Research Center, San José, CA 95120-6099, U.S.A.

Pierre WOLPER*
AT&T Bell Laboratories, Murray Hill, NJ 07974, U.S.A.

Abstract. The problem of complementing Büchi automata arises when developing decision procedures for temporal logics of programs. Unfortunately, previously known constructions for complementing Büchi automata involve a doubly exponential blow-up in the size of the automaton. We present a construction that involves only an exponential blow-up. We use this construction to prove a polynomial space upper bound for the propositional temporal logic of regular events and to prove a complexity hierarchy result for quantified propositional temporal logic.

1. Introduction

For many years, logics of programs were tools for reasoning about the input/output behavior of programs. When dealing with concurrent or nonterminating process (like operating systems) there is, however, a need to reason about infinite computation paths. These are the sequences of states that the computation goes through. In the propositional case they can be viewed as infinite sequences of propositional truth assignments. In [14], temporal logic was proposed to reason about such sequences. Later it was incorporated into the process logics of [7, 12, 25].

Recent works [20, 28] established a close relationship between temporal logic and the theory of ω-regular languages. The ω-regular languages are the analogue of the regular languages, but defined on infinite words rather than finite words. The notion of ω-regularity is robust and has a well-developed theory [4, 5, 23]. There are several characterizations of ω-regular languages, one of which is by Büchi automata [2]. A Büchi automaton is a finite automaton operating on infinite words. An infinite word is accepted by a Büchi automaton iff there is some run of the automaton on that word in which some state from a designated set of states appears infinitely often.

In [20, 28] several temporal logics are shown to have exactly the expressive power of Büchi automata; in other words, the class of sets of sequences described by those

* Present affiliation: Institut Montefiore, Université de Liège au Sart-Tilman, 4000 Liège, Belgium.

logics coincides with the class of ω-regular languages. One method to decide satisfiability for these logics is to build a Büchi automaton that accepts exactly the strings satisfying the formula. Since these logics are closed under negation, building this automaton involves complementing Büchi automata. That is, given a Büchi automaton A, one has to find a Büchi automaton $\bar{A}$ such that $L_\omega(\bar{A}) = \Sigma^\omega - L_\omega(A)$, where $L_\omega(A)$ denotes the language of infinite words accepted by A.

The complementation problem for Büchi automata was first studied by Büchi [2]. He showed that his automata are indeed closed under complementation. His proof, however, was not explicitly constructive. Later on, several explicit constructions were given [3, 4, 9, 19]. All these constructions, however, involve at least a doubly exponential blow-up. That is, there is a constant $c > 1$ such that if A has n states, then $\bar{A}$ has at least c^{c^n} states. This blow-up is very expensive computationally and causes the decision procedures using the complementation of Büchi automata to be highly inefficient. For example, the decision procedure described in [28] for the temporal logic ETL_r runs in exponential space, while the known lower bound for this logic is PSPACE.

In this paper we re-examine the complementation problem for Büchi automata. We prove, using Büchi's original ideas [2], that Büchi automata can be complemented with only an exponential blow-up. We then use the construction to show that the universality problem for Büchi automata, i.e., the problem whether, for a given Büchi automaton A, we have $L_\omega(A) = \Sigma^\omega$, is PSPACE-complete. (The analogous result for finite automata on finite words was proven by Meyer and Stockmeyer [11].)

These results turn out to be very useful in deciding satisfiability for various temporal logics. We first reconsider ETL_r, an extended temporal logic that directly reasons about ω-regular events [28]. As mentioned earlier, the best known decision procedure for this logic runs in exponential space. Using our results about Büchi automata, we improve the upper bound for ETL_r to polynomial space, which matches the lower bound.

We then turn to QPTL, a quantified propositional temporal logic [20]. While this logic has the same expressive power as ETL_r, its complexity is nonelementary, since S1S, the second-order theory of one successor, which is known to be nonelementary [10], is easily reducible to QPTL. Using our result and Büchi automata we prove that the class of satisfiable QPTL formulas in prenex normal form starting with an existential quantifier and with k alternations of quantifiers is complete for $\text{NSPACE}(\exp^k n)$ (i.e., a stack of k exponentials). We believe that this result is of general theoretical interest since QPTL is the first nonelementary logic we know of, where each alternation of quantifiers increases the space complexity by exactly one exponential.

2. Büchi automata and their complementation problem

A *Büchi automaton* is a nondeterministic finite automaton on infinite words. Formally, it is a tuple $A = (\Sigma, S, \rho, S_0, F)$, where Σ is an alphabet, S is a set of

states, $\rho: S \times \Sigma \to 2^S$ is a nondeterministic transition function, $S_0 \subseteq S$ is a set of starting states, and $F \subseteq S$ is a set of designated states. A *run* of A over an infinite word $w = a_1 a_2 \ldots$ is a sequence $s_0, s_1, \ldots$, where $s_0 \in S_0$, and $s_i \in \rho(s_{i-1}, a_i)$ for all $i \geq 1$. A run $s_0, s_1, \ldots$ is *accepting* if there is some designated state that repeats infinitely often, i.e., for some $s \in F$ there are infinitely many i's such that $s_i = s$. The infinite word w is *accepted* by A if there is an accepting run of A over w. The set of infinite words accepted by A is denoted $L_\omega(A)$. (Note that A can also be viewed as an automaton on *finite* words. The set of finite words accepted by A is denoted $L(A)$.)

We consider the problem of complementing Büchi automata. That is, given a Büchi automaton A that accepts the language $L_\omega(A)$, we want to construct an automaton $\bar{A}$ that accepts the language $\overline{L_\omega(A)} = \Sigma^\omega - L_\omega(A)$. This problem was first studied by Büchi [2]. He showed that his automata are indeed closed under complementation. His proof, however, was not explicitly constructive. Later on, explicit versions of Büchi's original proof were given [3, 19]. These explicit constructions, however, involve a doubly exponential blow-up. That is, there is a constant $c > 1$ such that if A has n states, then $\bar{A}$ has c^{c^n} states. Furthermore, this blow-up occurs whenever the construction is invoked, regardless of A's structure. Other constructions for complementing Büchi automata were given by McNaughton [9] and by Rabin [15] (see also [4]). McNaughton's construction is also doubly exponential, while Rabin's construction is nonelementary.

Here we describe a construction, based on Büchi's original proof, which, given a Büchi automaton A with n states, yields a Büchi automaton with $O(16^{n^2})$ states that accepts the complement of $L_\omega(A)$.

2.1. A generalized subset construction

To complement a Büchi automaton $A = (\Sigma, S, \rho, S_0, F)$, we first build a family $\{\tilde{A}_i\}$ of deterministic automata on finite words that captures the essential behavior of the automaton. The behavior that we are trying to capture is as follows: given a finite nonempty word x and two states $u, v \in S$,

(1) is there a run of A on x starting with u and ending with v?

(2) is there a run of A on x starting with u, ending with v, and containing some state in F?

To construct the automata $\{\tilde{A}_i\}$, we use a construction that can be viewed as a generalization of Rabin and Scott's *subset construction* [17].

Let $S = \{s_1, \ldots, s_n\}$ be the set of states of A. Define $S' = S \times \{0, 1\}$ and $S^* = (2^{S'})^n$. S^* has m states, denoted $p_1, \ldots, p_m$, where $m = 4^{n^2}$. Intuitively, a state in S^* is an n-tuple of sets of states of S labeled by 0 or 1. We need an n-tuple of sets rather than a single set, because we are trying to capture information about runs that can start in any state of S. The label on the state (0 or 1) indicates whether the run contains a state in F. The state set of the $\tilde{A}_i$'s is $\tilde{S} = S^* \cup \{p_0\}$, i.e., we add to S^* a special starting state p_0.

The deterministic transition function $\tilde{\rho}: \tilde{S} \times \Sigma \to S^*$ is defined as follows:

- $\langle S_1, \ldots, S_n \rangle = \tilde{\rho}(p_0, a)$ iff $S_i = \{\langle u, 0\rangle : u \in \rho(s_i, a)\} \cup \{\langle u, 1\rangle : u \in \rho(s_i, a) \cap F\}$;
- $\langle S_1, \ldots, S_n \rangle = \tilde{\rho}(\langle T_1, \ldots, T_n \rangle, a)$ iff

$$S_i = \{\langle u, 0\rangle : u \in \rho(v, a) \text{ for some } \langle v, j\rangle \in T_i\}$$
$$\cup \{\langle u, 1\rangle : u \in \rho(v, a) \text{ for some } \langle v, 1\rangle \in T_i\}$$
$$\cup \{\langle u, 1\rangle : u \in \rho(v, a) \cap F \text{ for some } \langle v, j\rangle \in T_i\}.$$

We now define $\tilde{A}_i$, $1 \leq i \leq m$, as the deterministic automaton on finite words $(\Sigma, \tilde{S}, \tilde{\rho}, p_0, \{p_i\})$. Let X_i be the set of finite words accepted by $\tilde{A}_i$, i.e., $X_i = L(\tilde{A}_i)$. The following lemma follows immediately from the fact that the $\tilde{A}_i$'s are deterministic.

Lemma 2.1. *$X_1, \ldots, X_m$ is a partition of Σ^+.*

The next lemma describes how the $\tilde{A}_i$'s capture the behavior of A.

Lemma 2.2. *Let $p_i = \langle S_1, \ldots, S_n \rangle$ and $x \in \Sigma^+$. Then $x \in X_i$ iff, for all pairs of states s_j, s_l of A,*

(1) *there is a run $u_0, \ldots, u_k$, $1 \leq k$, of A over x such that $u_0 = s_j$ and $u_k = s_l$ iff $\langle s_l, 0\rangle \in S_j$;*
(2) *there is a run $u_0, \ldots, u_k$, $1 \leq k$, of A over x such that $u_0 = s_j$, $u_k = s_l$, and $u_h \in F$ for some $1 \leq h \leq k$ iff $\langle s_l, 1\rangle \in S_j$.*

Corollary. *Suppose that $x_i \in X_i$, $x_j \in X_j$, and $x_i x_j \in X_k$. Then $X_i X_j \subseteq X_k$.*

In the next section we will show how the generalized subset construction can be used to complement Büchi automata. We believe that this construction is of general usefulness (cf. [24]).

2.2. The complementation construction

Consider now the languages $Y_{ij} = X_i X_j^\omega$ where $1 \leq i, j \leq m$. We say that Y_{ij} is *proper* if $X_i X_j \subseteq X_i$ and $X_j X_j \subseteq X_j$. We can prove the following results about these languages.

Lemma 2.3. $\Sigma^\omega = \bigcup \{Y_{ij} : Y_{ij} \text{ is proper}\}$.

Proof. The proof is based on Ramsey's Theorem and is a refinement of the proof of [2, Lemma 2.1]. Let us consider a infinite word $w = a_0 a_1 \ldots$. By Lemma 2.1, the word w, in combination with the languages X_i defined in the previous section, defines a partition of $\mathbb{N}$ into m sets $D_1, \ldots, D_m$ such that $i \in D_k$ iff $a_0 \ldots a_{i-1} \in X_k$. Clearly, there is some α such that D_α is infinite. Ramsey's Theorem tells us that, given a partition of all unordered pairs of elements from some infinite set Δ into

finitely many disjoint sets $C_1, \ldots, C_n$, there exists an infinite subset of Δ and a set C_k such that $\{a, b\} \in C_k$ for all pairs of distinct elements $a, b \in \Delta$. By Lemma 2.1, the word w, in combination with the languages X_i defined in the previous section, defines a partition of all pairs of elements in D_α into m sets $C_1, \ldots, C_m$ such that $\{i, j\} \in C_k$ iff $a_i \ldots a_{j-1} \in X_k$, where $i < j$. By Ramsey's Theorem, there exist an infinite subset $\{i_1, i_2, \ldots, i_q, \ldots\}$ of D_α and a set C_β such that $\{i_p, i_q\} \in C_\beta$ for all pairs i_p, i_q. This implies that the word w can be partitioned into

$$w_1 = a_0 \ldots a_{i_1-1}, \qquad w_2 = a_{i_1} \ldots a_{i_2-1}, \qquad w_3 = a_{i_2} \ldots a_{i_3-1}, \ldots,$$

where $w_1 \in X_\alpha$, and $w_i \in X_\beta$ for $i > 1$. Thus, $w \in Y_{\alpha\beta}$. Furthermore, we also have that $w_1 w_2 \in X_\alpha$ and $w_2 w_3 \in X_\beta$. By the corollary to Lemma 2.2, it follows that $Y_{\alpha\beta}$ is proper. □

Lemma 2.4. *For $1 \leq i, j \leq m$, either $L_\omega(A) \cap Y_{ij} = \emptyset$ or $L_\omega(A) \cap Y_{ij} = Y_{ij}$.*

Proof. We will prove that if one word $w \in Y_{ij}$ is in $L_\omega(A)$, then all words in Y_{ij} are in $L_\omega(A)$. Indeed, a word $w \in Y_{ij}$ can be decomposed into $w_i w_j^1 w_j^2 \ldots$ where $w_i \in X_i$ and, for all $k > 0$, $w_j^k \in X_j$. Consider a run of A on w and denote by s_i the state reached in that run at the end of w_i, s_j^1 the state reached at the end of $w_i w_j^1$, etc. The run is accepting iff the path taken through the automaton between s_j^k and s_j^{k+1} contains a state in F for infinitely many k. Now, any other word $y \in Y_{ij}$ can be decomposed similarly to w into $y_i y_j^1 y_j^2 \ldots$. By Lemma 2.2, there will also be a run of A on y such that the state reached at the end of y_i will be s_i, the state reached at the end of $y_i y_j^1$ will be s_j^1, etc. Moreover, there will be a path between s_j^k and s_j^{k+1} containing a state of F and labeled by y_j^k iff there is such a path labeled by x_j^k. Hence, if x is accepted by A, so is y. □

Lemma 2.5. $\overline{L_\omega(A)} = \bigcup \{Y_{ij} : Y_{ij} \cap L_\omega(A) = \emptyset$ *and* Y_{ij} *is proper*$\}$.

Proof. Immediate from Lemma 2.3 and 2.4. □

We now construct a Büchi automaton $\bar{A}$ that accepts $\overline{L_\omega(A)}$.

Theorem 2.6. *Let $A = (\Sigma, S, \rho, S_0, F)$ be a Büchi automaton with $|S| = n$. Then we can construct a Büchi automaton $\bar{A}$ with $O(16^{n^2})$ states such that $L_\omega(\bar{A}) = \overline{L_\omega(A)}$.*

Proof. From the automata $\tilde{A}_i$ and $\tilde{A}_j$ (each of size $m + 1 = 4^{n^2} + 1$), it is easy to build a Büchi automaton for Y_{ij} with $2m + 1$ states. Then, by Lemma 2.5, we only need to take the union of the automata for the languages Y_{ij} such that $Y_{ij} \cap L_\omega(A) = \emptyset$ and Y_{ij} is proper. Thus $\bar{A}$ will be a union of at most m^2 Büchi automata, each with at most $2m + 1$ states. The resulting automaton will thus have as many as $m^2(2m + 1) = O(64^{n^2})$ states. However, this construction is rather wasteful. Indeed, it contains as many as m^2 copies of the automata $\tilde{A}_i$'s. A more careful construction uses only $m + 1$ copies of these automata.

The idea of the more economical construction is to use a single copy of the set of states of the automata $\tilde{A}_i$ to recognize all the initial prefixes X_i of the languages Y_{ij}. Formally, we have the following. The automaton $\bar{A}$ is $(\Sigma, \bar{S}, \bar{\rho}, \{p_0\}, \bar{F})$. The state set $\bar{S}$ is $\tilde{S} \cup (\tilde{S} \times \{1, \ldots, m\})$. The designated state set $\bar{F}$ is $\{\langle p_0, i\rangle : 1 \leq i \leq m\}$. It remains to define the transition function $\bar{\rho}$.

- For $p_i \in \tilde{S}$, if $\tilde{\rho}(p_i, a) = p_j$, then $\bar{\rho}(p_i, a) = \{p_j\} \cup \{\langle p_0, l\rangle : Y_{jl} \cap L_\omega(A) = \emptyset$ and Y_{jl} is proper$\}$.
- For $\langle p_i, l\rangle \in (\tilde{S} \times \{1, \ldots, m\})$, if $\tilde{\rho}(p_i, a) = p_j$, then $\bar{\rho}(\langle p_i, l\rangle, a) = \{\langle p_j, l\rangle\}$ for $l \neq j$, and $\bar{\rho}(\langle p_i, l\rangle, a) = \{\langle p_0, l\rangle, \langle p_j, l\rangle\}$ for $l = j$.

We leave it to the reader to verify that $L_\omega(\bar{A}) = \overline{L_\omega(A)}$, and that $\bar{A}$ has at most $O(16^{n^2})$ states. □

Note that the above proof is not fully constructive since we did not specify how we can check for each Y_{ij} whether $Y_{ij} \cap L_\omega(A) = \emptyset$ or not.

Lemma 2.7. *Let $p_i = \langle S_1, \ldots, S_n\rangle$ and $p_j = \langle T_1, \ldots, T_n\rangle$.*

(1) *Y_{ij} is proper iff the following holds:*
   - *$\langle s_r, 0\rangle \in S_p$ iff there is a state $s_q \in S$ such that $\langle s_q, 0\rangle \in S_p$ and $\langle s_r, 0\rangle \in T_q$;*
   - *$\langle s_r, 1\rangle \in S_p$ iff there is a state $s_q \in S$ such that $\langle s_q, 0\rangle \in S_p$, $\langle s_r, 0\rangle \in T_q$, and either $\langle s_q, 1\rangle \in S_p$ or $\langle s_r, 1\rangle \in T_q$;*
   - *$\langle s_r, 0\rangle \in T_p$ iff there is a state $s_q \in S$ such that $\langle s_q, 0\rangle \in T_p$ and $\langle s_r, 0\rangle \in T_q$; and*
   - *$\langle s_r, 1\rangle \in T_p$ iff there is a state $s_q \in S$ such that $\langle s_q, 0\rangle \in T_p$, $\langle s_r, 0\rangle \in T_q$, and either $\langle s_q, 1\rangle \in T_p$ or $\langle s_r, 1\rangle \in T_q$.*

(2) *Assume Y_{ij} is proper. Then $Y_{ij} \subseteq L_\omega(A)$ iff there are states $s_p \in S_0$ and $s_q \in S$ such that $\langle s_q, 0\rangle \in S_p$ and $\langle s_q, 1\rangle \in T_q$.*

Proof. (1): By the corollary to Lemma 2.2, Y_{ij} is proper iff there are words $x_i \in X_i$ and $x_j \in X_j$ such that $x_i x_j \in X_i$ and $x_j x_j \in X_j$. The claim follows by the definition of X_i and X_j.

(2): It is easy to see that the condition is sufficient. We show that it is also necessary. Let $w \in Y_{ij} \subseteq L_\omega(A)$. That is, $w = w_1 w_2 \ldots$, where $w_1 \in X_i$, and $w_l \in X_j$ for $l > 1$. Consider an accepting run of A over w. Let s_p be the initial state of the run. Let t_l be the state reached after the prefix $w_1 \ldots w_l$ of w. Since S is finite, there is some $s_q \in S$ such that $s_q = t_l$ for infinitely many l's. It is easy to see that s_p and s_q satisfy the condition of the lemma. □

2.3. Decision problems

Two problems that we want to solve for Büchi automata are the *nonemptiness* and the *nonuniversality* problems; that is, given a Büchi automaton, determine whether there is some word it accepts (the nonemptiness problem), and whether there is some word it does not accept (the nonuniversality problem). The nonemptiness problem is studied in [27], where the following lemma and theorem are proved.

Lemma 2.8. *A Büchi automaton accepts some word iff there is a designated state of the automaton that is reachable from some initial state and is reachable from itself.*

Theorem 2.9. *The nonemptiness problem for Büchi automata is logspace complete for* NLOGSPACE.

We now turn to the nonuniversality problem for Büchi automata. Given a Büchi automaton A, the obvious way to solve this problem is to construct the automaton $\bar{A}$ and then use the algorithm for nonemptiness on $\bar{A}$. This gives an algorithm that uses exponential time and space as $\bar{A}$ is of size exponential in the size of A. However, as the fact that the nonemptiness problem is in NLOGSPACE indicates, it is possible to solve the nonemptiness of complement problems using only polynomial space. The argument is that it is not necessary to first build the whole automaton $\bar{A}$ before applying the algorithm for nonemptiness. In the rest of this paper, we will have to deal several times with the same type of construction: given an instance of a problem, construct a Büchi automaton that is exponentially big in the size of the problem, then determine if the Büchi automaton accepts some word. Each time, we will be able to show that the problem can be solved using only polynomial space. To avoid repeating the same argument several times, we will use the following lemma proved in [27].

Lemma 2.10. *Given a problem P and a Büchi automaton A which can be constructed from P, if*

(1) *the size of each state of A is polynomial in the size of P;*
(2) *it can be checked if a state is initial in space polynomial in the size of P; and*
(3) *it can be checked if a state is designated in space polynomial in the size of P; and*
(4) *each transition of A can be checked in space polynomial in the size of P (i.e., given states s and t of A and a letter $a \in \Sigma$, one can determine whether there is a transition from s to t labeled by a in polynomial space),*

then determining if A accepts some word can be done in space polynomial in the size of P.

Theorem 2.11. *The (non)-universality problem for Büchi automata is logspace complete in* PSPACE.

Proof. (*Hard for* PSPACE): We prove this using a reduction from the corresponding result for automata on finite words [11], that is, given an automaton $A = (\Sigma, S, \rho, S_0, F)$ where $\Sigma = \{a_1, \ldots, a_n\}$, determine whether $L(A) = \Sigma^+$ (without loss of generality we can assume that $\lambda \notin L(A)$). Define two new alphabets $\Sigma_1 = \{a_1^1, \ldots, a_n^1\}$ and $\Sigma_2 = \{a_1^2, \ldots, a_n^2\}$. Consider now the automaton $A_1 = (\Sigma_1, S, \rho_1, S_0, F)$, where $s' \in \rho_1(s, a_j^1)$ iff $s' \in \rho(s, a_j)$ for $s \in S$, and the automaton $A_2 = (\Sigma_2, S, \rho_2, S_0, F)$, where ρ_2 is defined analogously to ρ_1. The automata A_1 and A_2 thus recognize the image of $L(A)$ over the alphabets Σ_1 and Σ_2 respectively. We

now define a language L'_ω of infinite words over the alphabet $\Sigma_1 \cup \Sigma_2$ as follows:

$$L'_\omega = (L(A_1)L(A_2))^\omega \cup (L(A_2)L(A_1))^\omega \cup (L(A_1)L(A_2))^* L(A_1)^\omega$$
$$\cup (L(A_1)L(A_2))^* L(A_2)^\omega \cup (L(A_2)L(A_1))^* L(A_1)^\omega$$
$$\cup (L(A_2)L(A_1))^* L(A_2)^\omega.$$

It is easy to construct a Büchi automaton A' that recognizes L'_ω, i.e., $L_\omega(A') = L'_\omega$. The size of A' will be linear in the size of A and it can be constructed using logarithmic space. We now prove that A is universal (i.e., $L(A) = \Sigma^+$) iff A' is universal (i.e., $L'_\omega = \Sigma^\omega$).

First, let us assume that A is universal and prove that A' is universal. If A is universal, then $L(A_1)$ and $L(A_2)$ contain all nonempty words over Σ_1 and Σ_2 respectively. An infinite word over $\Sigma_1 \cup \Sigma_2$ is either entirely over Σ_1, or entirely over Σ_2, or consists of an alternation of finite words over Σ_1 and Σ_2, or, finally, consists of a finite alternation followed by an infinite word entirely over Σ_1 or Σ_2. The language L'_ω clearly takes all these cases into account and thus, if A is universal, A' will also be universal.

To prove that if A' is universal (over infinite words), then A is universal (over finite words), we will take an arbitrary nonempty finite word w and show that, under the assumption that A' is universal, that word is in $L(A)$. Given $w \in \Sigma^+$, let us consider the corresponding words w_1 and w_2 over Σ_1 and Σ_2 respectively. If A' is universal, then the infinite word $(w_1 w_2)^\omega$ is in L'_ω. Moreover, given the definition of L'_ω, it must be in the set described by $(L(A_1)L(A_2))^\omega$. Now, the only way this is possible is if w_1 is in $L(A_1)$ and w_2 in $L(A_2)$. Hence, w has to be in $L(A)$.

(*In* PSPACE): The fact that the universality problem for Büchi automata is in PSPACE follows from Theorem 2.6 and Lemmas 2.7 and 2.10. □

We note that our technique yields a polynomial space upper bound for the equivalence problem of Büchi automata. The previously known algorithm for this problem runs in exponential time and space [1].

3. Extended temporal logic

Temporal logic is a logic to reason about computations. It has been demonstrated to be very useful in the verification of concurrent and/or nonterminating processes [8, 13, 14]. Unfortunately, PTL, the standard propositional temporal logic (see [6, 14]), cannot express all regular properties. For example, Wolper has shown that PTL cannot express the property 'the proposition p holds at least in every other state of the computation' [30]. To remedy this deficiency, Wolper introduced an extension of temporal logic that incorporates nondeterministic finite automata as connectives. Clearly, regular properties can be expressed in the extended logic.

In [27, 28] three different versions of Wolper's extension were defined and studied further. The difference between the three versions is the type of acceptance conditions used for the finite automata defining the connectives. The three types of acceptance are *finite acceptance* (some prefix is accepted by the standard notion of acceptance for finite words), *looping acceptance* (the automaton has some infinite run over the word) and *repeating acceptance* (the automaton has a Büchi acceptance condition). These acceptance conditions give rise to three logics: $\mathrm{ETL_f}$, $\mathrm{ETL_l}$, and $\mathrm{ETL_r}$, correspondingly.

These logics all have the same expressive power. Nevertheless, while there is a linear translation from $\mathrm{ETL_f}$ and $\mathrm{ETL_l}$ to $\mathrm{ETL_r}$, the best known translation from $\mathrm{ETL_r}$ to $\mathrm{ETL_f}$ or $\mathrm{ETL_l}$ is doubly exponential [27]. This suggests that $\mathrm{ETL_r}$ is more succinct than $\mathrm{ETL_f}$ and $\mathrm{ETL_l}$. Moreover, while the decision problems for $\mathrm{ETL_f}$ and $\mathrm{ETL_l}$ are PSPACE-complete, the decision procedure for $\mathrm{ETL_r}$ presented in [28] required exponential space. Nevertheless, using our new results in Büchi automata, we will now show that the decision problem for $\mathrm{ETL_r}$ is also in PSPACE, hence, it is PSPACE-complete. Note, however, that the decision procedures for $\mathrm{ETL_f}$ and $\mathrm{ETL_l}$ require space $O(n^2)$, while the decision procedure for $\mathrm{ETL_r}$ requires space $O(n^4)$.

3.1. *Definition of the logic*

Formulas of $\mathrm{ETL_r}$ are built from a set of atomic propositions P by means of

- Boolean connectives, and
- automata connectives: every Büchi automaton $A=(\Sigma, S, \rho, S_0, F)$, where $\Sigma=\{a_1, \ldots, a_l\}$, is considered as an l-ary temporal connective. That is, if $f_1, \ldots, f_l$ are formulas, then so is $A(f_1, \ldots, f_l)$.

A structure for our logic in an infinite sequence of truth assignments, i.e., a function $\pi: \mathbb{N} \to 2^P$ that assigns truth values to the atomic propositions in each state. We use π^i to denote the ith 'tail' of π, i.e., $\pi^i(k)=\pi(k+i)$. We now define satisfaction of formulas by induction (satisfaction of a formula f by a structure π is denoted $\pi \models f$):

- $\pi \models f_1 \wedge f_2$ iff $\pi \models f_1$ and $\pi \models f_2$;
- $\pi \models \neg f$ iff not $\pi \models f$;
- $\pi \models A(f_1, \ldots, f_l)$ where $A=(\Sigma, S, \rho, S_0, F)$, iff there exists an infinite word $w=a_{i_0}a_{i_1}\ldots$ over Σ, accepted by A, such that $\pi^j \models f_{i_j}$ for all $j \geq 0$. (Intuitively, the transitions of A are labeled by the formulas $f_1, \ldots, f_l$, and $A(f_1, \ldots, f_l)$ is satisfied by π if there exists an accepting run of the automaton such that all the labels are satisfied by the corresponding suffixes of π.)

3.2. *Decision procedure for* $\mathrm{ETL_r}$

To give a PSPACE decision procedure for $\mathrm{ETL_r}$, we first need to introduce the notion of the closure of an $\mathrm{ETL_r}$ formula f, denoted $\mathrm{cl}(f)$. It is defined as follows (where we identify $\neg\neg g$ with g):

- $f \in \mathrm{cl}(f)$;

- $f_1 \wedge f_2 \in \mathrm{cl}(f) \rightarrow f_1, f_2 \in \mathrm{cl}(f)$;
- $\neg f_1 \in \mathrm{cl}(f) \rightarrow f_1 \in \mathrm{cl}(f)$;
- $f_1 \in \mathrm{cl}(f) \rightarrow \neg f_1 \in \mathrm{cl}(f)$;
- $A(f_1, \ldots, f_l) \in \mathrm{cl}(f) \rightarrow f_1, \ldots, f_l \in \mathrm{cl}(f)$.

When defining the length of a formula, we take the size of an automaton connective $A = (\Sigma, S, \rho, S_0, F)$ to be equal to $|S|+1$ (the "+1" is for technical reasons). For an $\mathrm{ETL_r}$ formula f, the size of $\mathrm{cl}(f)$ can easily be seen to be at most $2n$ where n is the length of f.

To establish a decision procedure for $\mathrm{ETL_r}$, we reduce the satisfiability problem to the emptiness problem for Büchi automata over the alphabet $2^{\mathrm{cl}(f)}$. To this end we extend the sequence $\pi : \mathbb{N} \rightarrow 2^P$ to a sequence $\psi : \mathbb{N} \rightarrow 2^{\mathrm{cl}(f)}$ in a natural way: for every $i \in \mathbb{N}$ and every formula $g \in \mathrm{cl}(f)$, we have that $g \in \psi(i)$ iff π^i satisfies g. Sequences that correspond to models satisfy some special properties.

A *Hintikka sequence* for an $\mathrm{ETL_r}$ formula f is a sequence $\psi : \mathbb{N} \rightarrow 2^{\mathrm{cl}(f)}$ that satisfies the following conditions:

(1) $f \in \psi(0)$; and, for all elements $i \in \mathbb{N}$,

(2) $g \in \psi(i)$ iff $\neg g \notin \psi(i)$;

(3) $g_1 \wedge g_2 \in \psi(i)$ iff $g_1 \in \psi(i)$ and $g_2 \in \psi(i)$;

(4) if $A(f_1, \ldots, f_l) \in \psi(i)$, where $A = (\Sigma, S, \rho, S_0, F)$, then there exists an infinite word $w = a_{j_0} a_{j_1} \ldots$ over Σ, accepted by A, such that $f_{j_k} \in \psi(i+k)$ for all $k \geq 0$;

(5) if $\neg A(f_1, \ldots, f_l) \in \psi(i)$, then there is no infinite word $w = a_{j_0} a_{j_1} \ldots$ over Σ, accepted by A, such that $f_{j_k} \in \psi(i+k)$ for all $k \geq 0$.

Lemma 3.1. *An* $\mathrm{ETL_r}$ *formula f has a model iff it has a Hintikka sequence.*

Proof. Given a model π for a formula f, its natural extention to $\mathbb{N} \rightarrow 2^{\mathrm{cl}(f)}$ is a Hintikka sequence. Given a Hintikka sequence ψ for a formula f, its *projection* π on P is a model for f (The projection $\psi|_P : \mathbb{N} \rightarrow 2^P$ of ψ on P is defined by $\psi|_P(i) = \psi(i) \cap P$.) Indeed, it can be shown by a simple induction on the structure of the formulas that, for each formula $g \in \psi(i)$, the sequence π^i satisfies g. Thus, as condition (1) requires that $f \in \psi(0)$, a Hintikka sequence defines a model for f. □

The next step in obtaining a decision procedure for $\mathrm{ETL_r}$ is to construct a Büchi automaton that accepts exactly the Hintikka sequences for a formula. To do this, we will actually build three automata. The *local automaton* A_{L} will check Hintikka conditions (1)–(3), the *positive automaton* A_{P} will check Hintikka condition (4) and the *negative automaton* A_{N} will check Hintikka condition (5).

The local automaton

The local automaton is $A_{\mathrm{L}} = (2^{\mathrm{cl}(f)}, 2^{\mathrm{cl}(f)}, \rho_{\mathrm{L}}, N_f, 2^{\mathrm{cl}(f)})$. The state set and the alphabet are thus the collection of all sets of formulas in $\mathrm{cl}(f)$.

For the transition relation, we have that $s' \in \rho_{\mathrm{L}}(s, a)$ iff $a = s$ and:

- $g \in s$ iff $\neg g \notin s$,
- $g_1 \wedge g_2 \in s$ iff $g_1 \in s$ and $g_2 \in s$.

The set of starting states N_f consists of all sets s such that $f \in s$. Clearly, A_L accepts precisely the sequences that satisfy Hintikka conditions (1)-(3).

The positive automaton

The positive automaton is actually the result of taking the intersection of a collection of automata, one for each formula of the form $A(f_1, \ldots, f_l)$ in $\mathrm{cl}(f)$. We will now describe how to build each of these automata. We need to construct an automaton defined over $2^{\mathrm{cl}(f)}$ that will run A for each j such that $A(f_1, \ldots, f_l) \in \psi(j)$. We will build this automaton from A in several steps.

Let $A = (\Sigma, S, \rho, S_0, F)$, where $\Sigma = \{a_1, \ldots, a_l\}$. First, we will transform A into an automaton A^0 over $2^{\mathrm{cl}(f)}$. The states of A^0 are the same as those of A and we have that $s' \in \rho^0(s, \boldsymbol{a})$ iff, for some $a_j \in \Sigma$, we have $s' \in \rho(s, a_j)$ and $f_j \in \boldsymbol{a}$.

We now need to transform the automaton A^0 so that it checks the sequence each time $A(f_1, \ldots, f_l)$ appears. To do this, we will use a construction similar to the 'flag construction' described in [4, 16]. Let us designate the number of states of A^0 by k ($|S| = k$). We add to S a state denoted by 0, which we call the *dormant state*, and we extend the transition relation as follows: for all $\boldsymbol{a} \in 2^{\mathrm{cl}(f)}$ such that $A(f_1, \ldots, f_l) \notin \boldsymbol{a}$, we have $0 \in \rho^0(0, \boldsymbol{a})$; and, for all $\boldsymbol{a}$ such that $A(f_1, \ldots, f_l) \in \boldsymbol{a}$, we have $s \in \rho^0(0, \boldsymbol{a})$ for each s, s' such that $s' \in S_0$ and $s \in \rho^0(s', \boldsymbol{a})$. Taking 0 to be the unique starting state, we get a new automaton A^1. Intuitively, A^1 stays in its dormant state until it sees an element containing $A(f_1, \ldots, f_l)$ and then starts running exactly as A^0.

Now, this is not enough as we need to run A^0 each time the formula $A(f_1, \ldots, f_l)$ appears. So, we need several copies of A^1. Fortunately, we only need as many copies as there are states in A^1 given that runs leading to the same state can be merged. Thus, we take $k+1$ copies of A^1 and combine them to form the automaton A^2. The states of A^2 are $k+1$-tuples of states of A^1, the initial state is 0^{k+1}. Thus the automaton A^2 has $(k+1)^{k+1}$ states. The transition relation ρ^2 is defined as follows: $(s'_1, \ldots, s'_{k+1}) \in \rho^2((s_1, \ldots, s_{k+1}), \boldsymbol{a})$ iff for each $1 \leq j \leq k+1$ either

- $s'_j \in \rho^1(s_j, \boldsymbol{a})$ and, for every $i < j$, $s'_i \neq s'_j$, or
- $s'_j = 0$ and there is an $i < j$ such that $s'_i \in \rho^1(s_j, \boldsymbol{a})$.

The only thing we still need to do is giving acceptance conditions for the automaton A^2. Recall that the states of A^2 are $k+1$ tuples of elements of $S \cup \{0\}$. Given a computation of A^2, its projections on the coordinates $1, \ldots, k$ represent computations of A^1 that check that condition (4) is satisfied for the occurrences of $A(f_1, \ldots, f_l)$. Consider now a computation that starts on coordinate j to check for a specific occurrence of $A(f_1, \ldots, f_l)$. Occasionally, this computation gets merged with another one on some coordinate i, $i < j$, in which case coordinate j goes into the state 0. Eventually, this computation reaches a coordinate j_0, $j_0 < j$, that it never leaves. If this computation does not accept, then the set of states occurring in it infinitely often is disjoint from $F \cup \{0\}$ (F is the designated set of A). Thus, the acceptance condition we need is that for each $1 \leq j \leq k+1$ some state containing 0 or an element of F in its jth position appears infinitely often in the computation of A^2. Notice that this is not a Büchi acceptance condition. Indeed, we need to

check that the set of infinitely-often appearing states in the computation of A^2 nontrivially intersects, not one set of states, but $k+1$ sets of states. Now we repeat this construction for each formula of the form $A(f_1, \ldots, f_l)$ in $\mathrm{cl}(f)$, and take the cross product of all these automata. The states of the product automaton are r-tuples of states of the various automata A^1's, for some $r \leq n$, where n is the length of f. Thus, this automaton has $O(n^n)$ states. This automaton, however, has r sets of designated states. As was shown in [26], such an automaton can be converted into a Büchi automaton whose size is r times the size of the original automaton. Thus A_P has $O(n^{n+1})$ states.

The negative automaton

The most straightforward way to deal with this case is to build, for each formula of the form $\neg A(f_1, \ldots, f_l)$, the complement of the automaton A and then apply the flag construction as in the preceding case. This, however, would lead to a doubly exponential blow-up and it is possible to be more efficient. We do that by first building an automaton that tries to find an occurrence of $\neg A(f_1, \ldots, f_l)$ for which Hintikka condition (5) is not satisfied, and then taking the complement of this automaton.

We start by constructing the automaton A^0 exactly as in the previous case. We then build an automaton A^1. The automaton A^1 has the same states as the automaton A^0 plus a dormant state 0. Its transition function is the one of A^0 extended as follows: for all $\boldsymbol{a} \in 2^{\mathrm{cl}(f)}$, we have $0 \in \rho^1(0, \boldsymbol{a})$; and, for all $\boldsymbol{a}$ such that $\neg A(f_1, \ldots, f_n) \in \boldsymbol{a}$, we have $s \in \rho^1(0, \boldsymbol{a})$ for each s, s' such that $s' \in S_0$ and $s \in \rho^0(s', \boldsymbol{a})$. Acceptance is defined exactly as for A. The automaton we have built stays in the dormant state until it sees an element containing $\neg A(f_1, \ldots, f_l)$ and then either stays in the dormant state or starts running exactly as A^0. The sequences it accepts are thus those in which $\neg A(f_1, \ldots, f_n)$ appears at some point and that satisfy $A(f_1, \ldots, f_l)$ from that point. This is exactly the complement of the set of sequences we are checking for. We now take A^2 to be the union of automata A^1 over all formulas of the form $\neg A(f_1, \ldots, f_l)$ in $\mathrm{cl}(f)$. A^2 has at most n states, where n is the length of f. Now A_N is taken to be the complement of A^2, using the construction we described in Section 2. A_N has $O(16^{n^2})$ states.

We now have the following proposition.

Proposition 3.2. *Let f be an* ETL_r *formula. Then, one can construct a Büchi automaton of size exponential in the length of f such that a sequence $\psi : \mathbb{N} \to 2^{\mathrm{cl}(f)}$ is accepted by that automaton iff ψ is a Hintikka sequence for f.*

Proof. Let A be the automaton that corresponds to the intersection of the local, positive, and negative automata for f. (By [4], if A_1 and A_2 are Büchi automata with m and n states respectively, then one can construct a Büchi automaton B with $O(mn)$ states such that $L_\omega(B) = L_\omega(A_1) \cap L_\omega(A_2)$.) This automaton, over the

alphabet $2^{\mathrm{cl}(f)}$, has $O(c^{n^2})$ states for some constant $c>1$ and it accepts precisely all Hintikka sequences for f. □

Before showing that we can construct an automaton that accepts precisely the sequences that satisfy f, we need some technical tools, which will also be useful in the next section. Given two alphabets Σ and Σ', we call a mapping $\pi: \Sigma \to \Sigma'$ a *projection* from Σ to Σ'. Given an infinite word $w = a_1 a_1 \ldots$ over Σ, $\pi(w) = \pi(a_1)\pi(a_2)\ldots$ is an infinite word over Σ'. Given a set $L \subseteq \Sigma^\omega$, then $\pi(L) = \{\pi(w): w \in L\}$.

Lemma 3.3. *Given a Büchi automaton A with n states and a projection $\pi: \Sigma \to \Sigma'$, there is a Büchi automaton A' with n states such that $L_\omega(A') = \pi(L_\omega(A))$.*

Proof. Let $A = (\Sigma, S, \rho, S_0, F)$. Define A' to be $(\Sigma', S, \rho', S_0, F)$, where ρ' is defined by $\rho'(s, a') = \{t: t \in \rho(s, a)$ for some $a \in \Sigma$ such that $\pi(a) = a'\}$. We leave it to the reader to verify that $L_\omega(A') = \pi(L_\omega(A))$. □

We call the automaton A' in the above lemma the *projection* of A on Σ'.

We are now ready to prove the desired result.

Theorem 3.4. *Let f be an* ETL_r *formula. Then, one can construct a Büchi automaton of size exponential in the length of f such that a sequence $\psi: \mathbb{N} \to 2^P$ is accepted by that automaton iff $\psi \models f$.*

Proof. Let A be the automaton given by Proposition 3.2. Consider the projection $\pi: 2^{\mathrm{cl}(f)} \to 2^P$, defined by $\pi(a) = a \cap P$. It follows from the proof of Lemma 3.1 that $\pi(L_\omega(A))$ is the set of sequences that satisfy f. The projection of A on 2^P is the desired automaton. □

Theorem 3.5. *The satisfiability problem for* ETL_r *is logspace complete in* PSPACE.

Proof. The hardness result follows easily from the hardness results in [21]. To prove that the problem is in PSPACE, it is sufficient to observe that the automata, A_L, A_P, and A_N satisfy the conditions of Lemma 2.10. Thus, the automaton corresponding to their intersection also satisfies the conditions of Lemma 2.10 and the problem of determining if this automaton accepts some word, which is equivalent by Proposition 3.1 to determining if the formula is satisfiable, is in PSPACE. □

The above proof shows that the decision procedure for ETL_r requires nondeterministic space $O(n^2)$ and, consequently, deterministic space $O(n^4)$. In contrast, the

decision procedures for ETL_f and ETL_r requires nondeterministic space $O(n)$ and, consequently, deterministic space $O(n^2)$ [27].

4. Quantified propositional temporal logic

In the previous section, we proved a result about one possible extension of temporal logic. There are other ways to extend temporal logic, one of which is to introduce quantification over propositions. This extension, quantified propositional temporal logic (QPTL), was described in [20]. It turns out that it has exactly the same expressive power as the extended temporal logic we studied in Section 3 [28, 29]. Nevertheless, the decisions problems for these logics have drastically different complexities.

Formulas of QPTL are built from a set of atomic propositions P using

- Boolean connectives;
- the temporal operators X (next) and F (eventually). We will also use G as an abbreviation for $\neg F \neg$;
- quantification over propositions (i.e., if $f(p)$ is a formula, then so is $(\exists p)(f(p))$); we will also use $\forall$ as an abbreviation for $\neg\exists\neg$.

We will say that a QPTL formula is in *normal form* if it can be written as

$$(Q_1 p_1 Q_2 p_2 \ldots Q_k p_k)(f),$$

where each Q_i is either $\forall$ or $\exists$ and f is a quantifier-free formula. (Every QPTL formula is equivalent to a formula in normal form.) If Q_1 is $\exists$ and there are $k-1$ alternations of quantifiers, we say that the formula is in the set Σ_k^{QPTL}. If Q_1 is $\forall$ and there are $k-1$ alternations of quantifiers, we say that the formula is in the set Π_k^{QPTL}. For example, $(\exists p)(Gp)$ is in Σ_1^{QPTL}, and $(\forall p\, \exists q\, \exists r)(Gp \supset (Fq \wedge Fr))$ is in Π_2^{QPTL}.

QPTL formulas are interpreted over infinite sequence of truth assignments, i.e., functions $\pi : \mathbb{N} \to 2^P$ that assign truth values to the atomic propositions in each state. We use π^i to denote the ith 'tail' of π, i.e., $\pi^i(k) = \pi(k+i)$. We now inductively define satisfaction of formulas:

- for an atomic proposition p, $\pi \models p$ iff $p \in \pi(0)$;
- $\pi \models f_1 \wedge f_2$ iff $\pi \models f_1$ and $\pi \models f_2$;
- $\pi \models \neg f$ iff not $\pi \models f$;
- $\pi \models Xf$ iff $\pi^1 \models f$;
- $\pi \models Ff$ iff there is an $i \geq 0$ such that $\pi^i \models f$;
- $\pi \models (\exists p)(f)$ iff there is some π' that agrees with π except for the proposition p and such that $\pi' \models f$

Before stating our complexity results on QPTL, we need one definition. Let us define $g_c(k, n)$ as follows:

$$g_c(0, n) = n, \qquad g_c(k+1, n) = c^{g(k,n)}$$

(i.e., $g_c(k, n)$ has a stack of k exponents). We use $g(k, n)$ to denote $g_2(k, n)$.

Let NSPACE($g(k, n)$) denote the class of languages accepted by a nondeterministic Turing machine in space $O(g_c(k, n))$ for some $c > 1$. Note that, for $k > 0$, the class NSPACE($g(k, n)$) is identical to its deterministic analogue SPACE($g(k, n)$). We prove the following theorem.

Theorem 4.1. *The satisfiability problem for* Σ_k^{QPTL}, $k \geq 1$, *is complete for* NSPACE($g(k-1, n)$).

We note that this result also holds for *weak* QPTL, in which all predicates are finite, i.e., they are eventually false forever. A result closely related to ours was proven by Robertson [18]. Robertson studied WS1S, the theory of natural numbers with successor with quantification over finite sets, which is equivalent to weak QPTL. He showed that Σ_k^{WS1S} is in NTIME($g(k+1, p(n))$) and is logspace hard for NTIME($g(k, p(n))$) (the "$p(n)$" denotes union over all polynomials p). By considering QPTL rather than S1S and by using our complementation result, we were able to close the gap between the lower and upper bounds, and also deal with quantification over infinite sets. We believe our result to be of general theoretical interest since QPTL is the first nonelementary logic we know of where each alternation increases the complexity by exactly one exponential.

4.1. Upper bounds

The proof of our upper bounds, will be based on the construction of a Büchi automaton that accepts exactly the sequences satisfying a Σ_k^{QPTL} formula. More precisely, we prove the following lemma.

Lemma 4.2. *There is a constant* $c > 1$ *such that, given a* Σ_k^{QPTL} *formula* f *of size* n, $k \geq 1$, *we can construct a Büchi automaton of size* $O(g_c(k, n))$ *that accepts exactly the sequences satisfying* f

Proof. The proof proceeds by induction.

(*Base step*, $k = 1$): Let P' be the set of all propositions in f, and let P'' be the set of free propositions in f. Note that $P'' \subseteq P'$. We apply the exponential construction described in [27, 28] to the quantifier-free part of f. This construction yields an automaton A over the alphabet $2^{P'}$ of size $O(\alpha^n)$ for some $\alpha > 1$. Consider the projection $\pi : 2^{P'} \to 2^{P''}$ defined by $\pi(a) = a \cap P''$. Clearly, $\pi(L_\omega(A))$ is the set of sequences that satisfy f. Thus, the projection of A on $2^{P''}$ is the desired automaton.

(*Inductive step*): We have to establish the result for formulas in Σ_k^{QPTL} knowing the result for formulas in $\Sigma_{k-1}^{\mathrm{QPTL}}$. First, notice that a formula in Σ_k^{QPTL} can be written as $(\exists p_1, \ldots, p_1)(\neg f_1)$, where f_1 is a formula in $\Sigma_{k-1}^{\mathrm{QPTL}}$. Now, we inductively build an automaton for f_1 and then we construct an automaton for f by complementing that automaton and projecting it, as above, on a smaller alphabet to eliminate the existentially quantified propositions. A simple analysis gives the desired bound on the size of the automaton. □

We can now establish our upper bounds.

Theorem 4.3. *Satisfiability for formulas of* Σ_k^{QPTL}, $k \geq 1$, *is in* NSPACE$(g(k-1, n))$.

Proof. Let us first consider the case $k=1$. This is equivalent to showing that satisfiability for quantifier-free temporal logic formulas can be tested in nondeterministic linear space. This was done in [21].

In the case $k>1$, we have a formula f of the form $(\exists p_1, \ldots, p_l)(\neg f_1)$, where $f_1 \in \Sigma_{k-1}^{\text{QPTL}}$. By Lemma 4.2, we know that we can construct an automaton for f_1 of size $O(g_c(k-1, n))$. Now, to check if the formula f is satisfiable, it is sufficient to check that there is some word not accepted by the automaton for f_1. By Theorem 2.11, this can be done in space polynomial in the size of the automaton for f_1 and hence in NSPACE$(g(k-1, n))$. □

4.2. Lower bounds

We will now prove the lower bound for Σ_k^{QPTL}. We use the method of [22]. The first step is to construct 'yardsticks' of nonelementary length. To do this, we show how to construct a formula $\varphi_{c,k,n}(p, q)$, which asserts that p and q are true exactly once and are separated by a distance greater than $g_c(k, n)$. Let us define $h_c(k, n)$ as follows:

$$h_c(0, n) = n, \qquad h_c(k+1, n) = h_c(k, n)2^{h_c(k,n)}.$$

It is easy to see that $h_c(k, n) \geq g_c(k, n)$. We now prove the following lemma.

Lemma 4.4. *Given* $k \geq 0$ *and* $n \geq 1$, *one can construct in space* $O(\log n)$ *a formula* $\varphi_{c,k,n}(p, q) \in \Sigma_k^{\text{QPTL}}$ *of length* $O(k+n)$ *such that if* $\pi \models \varphi_{c,k,n}(p, q)$, *then*

(a) *p and q are each true at exactly one point; and*

(b) *if* $\pi^i \models p$ *and* $\pi^j \models q$, *then* $j = i + h_c(k, n)$.

Proof. We give the argument for $c=2$. The modification for arbitrary $c>1$ is straightforward. We write $\varphi_{k,n}$ instead of $\varphi_{c,k,n}$. The proof will proceed by induction. We show how to construct $\varphi_{0,n}$ and then show how to construct $\varphi_{k+1,n}$ when given $\varphi_{k,n}$.

(*Base step*, $k=0$): First note that condition (a) of Lemma 4.4 can be stated by

$$Fp \wedge G(p \supset XG\neg p) \wedge Fq \wedge G(q \supset XG\neg q). \tag{1}$$

We force p and q to be separated by a distance $h(0, n)$ by the formula

$$G(p \supset X^n q). \tag{2}$$

We can take $\varphi_{0,n}$ to be the conjunction of (1) and (2). Notice that the length of $\varphi_{0,n}$ is $O(n)$.

(*Inductive step*): We now show how to construct $\varphi_{k+1,n}$ given $\varphi_{k,n}$ for $k \geqslant 0$. Intuitively, to define $\varphi_{k+1,n}(p, q)$, we will encode a counter with $h(k, n)$ bits by the value of a proposition b over consecutive blocks of $h(k, n)$ states. To describe the counter, we will use a proposition r that is true at the points where p or q are true and at intervals of $h(k, n)$ (see Fig. 1).

We first need to state condition (a). This is done by (1) as in the previous case. Next, we state the requirements on p, q, and r.

$$\mathrm{G}(p \vee q \supset r) \wedge \forall st(\varphi_{k,n}(s, t) \supset [\mathrm{G}(s \supset \mathrm{F}(r \wedge \mathrm{F}t)) \wedge \mathrm{G}((r \wedge \mathrm{F}t \wedge \neg \mathrm{F}s) \supset \neg \mathrm{XF}(r \wedge \mathrm{F}t))]). \tag{3}$$

The last clause requires that r is true at points separated by $h(k, n)$ by stating that, between every pair of points s and t at a distance of $h(k, n)$, the proposition r is true exactly once.

Using r, we will be able to state that on successive blocks of $h(k, n)$ states between p and q, the proposition b encodes a binary counter modulo $h(k, n)$ that starts at 0 and finishes at $2^{h(k,n)} - 1$ (we consider the leftmost bit of a number to be its least significant bit). We first need to state that the counter has value 0 immediately following p, value $2^{h(k,n)} - 1$ immediately preceding q and is never 0 between these points. We start by expressing that its value is 0 following p.

$$\forall z_1[\mathrm{F}z_1 \wedge \mathrm{G}(z_1 \supset \mathrm{XG}\neg z_1) \wedge \mathrm{G}(p \supset (\mathrm{F}z_1 \wedge \neg \mathrm{XF}(r \wedge \mathrm{F}z_1)))] \supset \mathrm{G}(z_1 \supset \neg b). \tag{4}$$

The first part of this formula states that z_1 is true at exactly one point that is between p and the second occurrence of r. We now express that all the bits of the counter are 1 in the last interval preceding q.

$$\mathrm{G}\{[\mathrm{XF}(r \wedge q) \wedge \neg \mathrm{XF}(r \wedge \mathrm{XF}q)] \supset b\}. \tag{5}$$

To state that the counter is never 0 in between p and q, we use

$$\forall z_2[\mathrm{F}z_2 \wedge \mathrm{G}(z_2 \supset (r \wedge \mathrm{XG}\neg z_2)) \wedge \mathrm{G}(p \supset \mathrm{XF}(r \wedge \mathrm{XF}z_2))] \supset \mathrm{F}[b \wedge \mathrm{F}z_2 \wedge \neg \mathrm{XF}(r \wedge \mathrm{XF}z_2)]. \tag{6}$$

The first part of this formula states that z_2 is true at exactly one point where r is true but is not one of the first two such points. The second part states that b should hold at some point not separated from z_2 by r.

Finally, we have to state that the value of each succeeding number is equal to the value of the preceding number incremented by 1. This can be expressed by

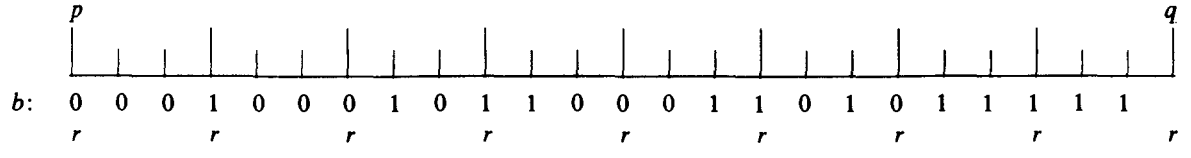

Fig. 1. $h(k, m) = 3$.

giving the relation between the values of the bits in corresponding positions of successive numbers. Such positions are those between points which are separated by a distance $h(k, n)$. Recall that if $\boldsymbol{a} = a_{n-1}, a_{n-2}, \ldots, a_0$ and $b = b_{n-1}, b_{n-2}, \ldots, b_0$ are two n-bit counters, then $\boldsymbol{b}$ is the successor of $\boldsymbol{a}$ (modulo 2^n) iff the following holds:

$$a_i = b_i \text{ iff } \exists r < i \text{ such that } a_r = 0.$$

This can be expressed by the following formula:

$$\forall st(\varphi_{k,n}(s, t) \supset [(\mathrm{G}(t \supset b) \equiv \mathrm{G}(s \supset b)) \equiv \mathrm{G}((r \wedge \mathrm{F}s \wedge \neg \mathrm{XF}(r \wedge \mathrm{F}s)) \supset \mathrm{F}(\neg b \wedge \mathrm{XF}s))]). \tag{7}$$

The formula $\varphi_{k+1,n}$ will then be the conjunction of (1), (3)–(7). By coalescing the quantifiers "$\forall st(\varphi_{k,n}(s, t)$" and converting the formula to normal form, we get a formula of the form:

$$(\exists rb)(\forall stz_1z_2)(f), \tag{8}$$

where the only quantifiers appearing in f are those appearing in $\varphi_{k,n}$. Given that these occurrences are all within the scope of exactly one negation, and that $\varphi_{k,n} \in \Sigma_k^{\mathrm{QPTL}}$, we have that $f \in \Pi_k^{\mathrm{QPTL}}$. Hence, $\varphi_{k+1,n} \in \Sigma_{k+1}^{\mathrm{QPTL}}$. Moreover, $\varphi_{k+1,n}$ satisfies the conditions of the lemma. Finally, it is easily seen that the length of $\varphi_{k,n}$ is $\mathrm{O}(k+n)$, and it can be obtained in space $\mathrm{O}(\log n)$.

To encode yardsticks of length $h_c(k, n)$ for $c > 2$ we use c-ary counters instead of binary counters. Instead of the proposition b that encodes a binary digit, we use several propositions that together encode a c-ary digit. The details are straightforward and left to the reader. □

We now prove our lower bounds, by encoding computations of Turing machines.

Theorem 4.5. *Every language in* $\mathrm{NSPACE}(g(k-1, n))$, $k \geq 1$, *is logspace reducible to the satisfiability problem for* Σ_k^{QPTL}.

Proof. We give the proof for $k \geq 2$. The case $k = 1$ directly follows from the results in [21]. We show that, given a $h_c(k-1, n)$-space bounded nondeterministic Turing machine M (and hence, a $g_c(k-1, n)$-space bounded Turing machine) and given an input y of length n, we can construct, using space $\mathrm{O}(\log n)$, a QPTL formula that is satisfiable iff M accepts y. First, we give some definitions concerning Turing machines.

A nondeterministic Turing machine is a tuple $M = (Q, \Gamma, \delta, q_1, q_A)$ where Q is the set of states, Γ is the alphabet, $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{\text{left,right}\}}$ is the transition function, q_1 is the initial state and q_A is the accepting state. We consider $h_c(k-1, n)$-space bounded Turing machines, that is, machines that use at most $h_c(k-1, n)$ tape cells. We assume that M is defined such that all these tape cells are to the immediate right of the initial head position. An instantaneous description (ID) of M on input y of length n is a word of length $h_c(k-1, n)$ in $\Gamma^*(Q \times \Gamma)\Gamma^*$. The ID $\mu(q, a)\nu$

means that $\mu a \nu$ is written on the tape and that M is in state q scanning symbol a. The initial ID on input $y = y_1 \ldots y_n$ is

$$\mathrm{ID}_0(y) = (q_1, y_1)y_2 \ldots y_n \# ^{h_c(k-1,n)-n},$$

where $\#$ denotes the blank-tape symbol. If $\alpha = \alpha_1 \ldots \alpha_{h_c(k-1,n)}$ is an ID, then an ID $\beta = \beta_1 \ldots \beta_{h_c(k-1,n)}$ is a successor of α if it is obtained from α by one transition of M. Let $\Delta = \Gamma \cup (Q \times \Gamma)$. It is known [22] that there is a function $R_M : \Delta^3 \to 2^{\Delta^3}$ such that β is a successor of α iff

$$(\beta_{i-1}, \beta_i, \beta_{i+1}) \in R_M(\alpha_{i-1}, \alpha_i, \alpha_{i+1}) \qquad (*)$$

for all $1 < i < h_c(k-1, n)$. A computation of M on input y is a sequence of ID's $\mathrm{ID}_0(y)\mathrm{ID}_1 \ldots$ that starts with the initial ID and such that, for all $i \geq 1$, ID_i is a successor of ID_{i-1}. An accepting ID is one in which the state is the accepting state. A computation is accepting if it contains an accepting ID.

We now construct a formula f that is satisfiable iff the computation of M on an input y is accepting. To do this, we represent each ID by a block of $h_c(k-1, n)$ consecutive states. In such a block of states, each individual state represents one element of the ID. We use a set P_M of propositions containing an element p_a for each $a \in (Q \times \Gamma) \cup \Gamma$. The unique member of P_M that is true in a given state represents the symbol at that position in the ID. We will also use a proposition r that marks the beginning of each successive ID. The situation is described in Fig. 2.

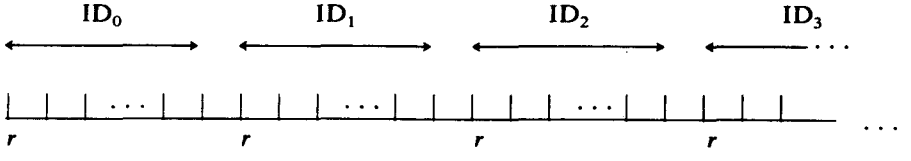

Fig. 2.

The formula f first needs to state that the proposition r behaves as desired. This is done with a statement similar to (3):

$$r \wedge \forall st(\varphi_{c,k-1,n}(s, t) \supset [\mathrm{G}(s \supset \mathrm{F}(r \wedge \mathrm{F}t)) \wedge \mathrm{G}((r \wedge \mathrm{F}t \wedge \neg \mathrm{F}s) \supset \neg \mathrm{XF}(r \wedge \mathrm{F}t))]). \qquad (9)$$

Now, we need to state that

(a) the first ID is $\mathrm{ID}_0(y)$;

(b) each ID is obtained by the previous one by a transition of M;

(c) an accepting ID eventually appears.

In these statements, we will use the notation $S = a$ to denote the formula that states that, among the propositions in P_M, only p_a is true. $S = a$ thus corresponds to a conjunction of p_a and the negation of the propositions representing other

symbols. Statement (a) is expressed by

$$S=(q_I, y_0) \wedge X(S=y_1 \wedge X(S=y_2 \wedge \cdots \wedge X(S=y_n)\cdots)$$
$$\wedge (\forall z)\{[X^{n+1}Fz \wedge G(z \supset XG\neg z) \wedge XG(r \supset G\neg z)] \supset G(z \supset (S=\not=))\}. \tag{10}$$

To express (b), we state that every tuple of elements of successive ID's separated by a distance $h_c(k-1, n)$ satisfies the relation (*).

$$\bigwedge_{abc\in(Q\times\Gamma)\cup\Gamma} \forall st[\varphi_{c,k-1,n}(s, t) \wedge G(s \supset X\neg r) \wedge G(s \supset XX\neg r)$$
$$\wedge G(s \supset S=a) \wedge G(s \supset XS=b) \wedge G(s \supset XXS=c)]$$
$$\supset \bigvee_{(d,e,f)\in R_M(a,b,c)} [G(t \supset S=d) \wedge G(t \supset XS=e) \wedge G(t \supset XXS=f)]. \tag{11}$$

To state (c), it is enough to state that a final state eventually appears

$$\bigvee_{a\in\Gamma} F(S=(a, q_A)). \tag{12}$$

The formula f is then the conjunction of (9)-(12). If we convert it to normal form, we get a formula of the form

$$(\exists r)(\forall stz)f', \tag{13}$$

where the only quantifiers appearing in f' are those appearing in $\varphi_{c,k-1,n}$. Given that these occurrences are all within the scope of exactly one negation, and that $\varphi_{c,k-1,n} \in \Sigma_{k-1}^{QPTL}$, we have that $f' \in \Pi_{k-1}^{QPTL}$. Hence $f \in \Sigma_k^{QPTL}$. Moreover, by construction, f is satisfiable iff y is accepted by M and it can be obtained using space $O(\log n)$. □

Acknowledgment

We would like to thank the referees for many helpful suggestions.

References

[1] H. Alaiwan, Equivalence of infinite behavior of finite automata, *Theoret. Comput. Sci.* **31** (1984) 297-306.

[2] J.R. Büchi, On a decision method in restricted second-order arithmetic, in: *Proc. Internat. Congr. Logic, Methodology and Philosophy of Science 1960* (Stanford University Press, 1962) 1-12.

[3] J.R. Büchi, The monadic theory of ω_1, in: *Decidable Theories II*, Lecture Notes in Mathematics **328** (Springer, Berlin, 1973) 1-127.

[4] Y. Choueka, Theories of automata on ω-tapes: a simplified approach, *J. Comput. System Sci.* **8** (1974) 117-141.

[5] S. Eilenberg, *Automata, Languages and Machines, Vol. A* (Academic Press, New York, 1974).

[6] D. Gabbay, A. Pnueli, S. Shelah and J. Stavi, The temporal analysis of fairness, *Proc. 7th ACM Symp. on Principles of Programming Languages*, Las Vegas (1980) 163-173.

[7] D. Harel, D. Kozen and R. Parikh, Process logic: expressiveness, decidability, completeness, *J. Comput. System Sci.* **25** (1982) 144–170.

[8] Z. Manna and A. Pnueli, Verification of concurrent programs: the temporal framework, in: R.S. Boyer and J.S. Moore, eds., *The Correctness Problem in Computer Science* (Academic Press, New York/London, 1981) 215–273.

[9] R. McNaughton, Testing and generating infinite sequences by a finite automaton, *Inform. and Control* **9** (1966) 521–530.

[10] A.R. Meyer, Weak monadic second-order theory of successor is not elementary recursive, *Proc. Logic Colloquium*, Lecture Notes in Mathematics **453** (Springer, Berlin, 1975) 132–154.

[11] A.R. Meyer and L.J. Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential time, *Proc. 13th IEEE Symp. on Switching and Automata Theory*, Long Beach (1972) 125–129.

[12] H. Nishimura, Descriptively complete process logic, *Acta Inform.* **14** (1980) 359–369.

[13] S. Owicki and L. Lamport, Proving liveness properties of concurrent programs, *Trans. ACM* **4** (1982) 455–495.

[14] A. Pnueli, The temporal logic of concurrent programs, *Theoret. Comput. Sci.* **13** (1981) 45–60.

[15] M.O. Rabin, Decidability of second-order theories and automata on infinite trees, *Trans. AMS* **141** (1969) 1–35.

[16] M.O. Rabin, Weakly definable relations and special automata, in: Y. Bar-Hillel, ed., *Proc. Symp. Mathematical Logic and Foundations of Set Theory* (North-Holland, Amsterdam, 1970) 1–23.

[17] M.O. Rabin and D. Scott, Finite automata and their decision problems, *IBM J. Res. & Dev.* **3** (1959) 114–125.

[18] E.L. Robertson, Structure of complexity in the weak monadic second-order theory of the natural numbers, *Proc. 6th ACM Symp. on Theory of Computing*, Seattle (1974) 161–171.

[19] D. Siefkes, *Decidable Theories I—Büchi's Monadic Second-Order Successor Arithmetics*, Lecture Notes in Mathematics **120** (Springer, Berlin, 1970).

[20] A.P. Sistla, Theoretical issues in the design and verification of distributed systems, Ph.D. Thesis, Harvard University, 1983.

[21] A.P. Sistla and E.M. Clarke, The complexity of propositional linear time logics, *J. ACM* **32** (1985) 733–749.

[22] L.J. Stockmeyer, The complexity of decision problems in automata theory and logic, Ph.D. Dissertation, Tech. Rept. MAC MIT TR-133, M.I.T., 1974.

[23] B.A. Trakhtenbrot and Y.M. Barzdin, *Finite Automata Behavior and Synthesis* (North-Holland, Amsterdam, 1973).

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

[25] M.Y. Vardi and P. Wolper, Yet another process logic, in: *Logics of Programs*, Lecture Notes in Computer Science **164** (Springer, Berlin, 1983) 501–512.

[26] M.Y. Vardi and P. Wolper, Automata-theoretic techniques for modal logics of programs, *J. Comput. System Sci.* **32** (1986) 183–221.

[27] M.Y. Vardi and P. Wolper, Reasoning about infinite computation paths, to appear.

[28] P. Wolper, M.Y. Vardi and A.P. Sistla, Reasoning about infinite computation paths, *Proc. 24th IEEE Symp. on Foundations of Computer Science*, Tucson (1983) 185–194.

[29] P. Wolper, Synthesis of communicating processes from temporal logic specifications, Ph.D. Thesis, Stanford University, 1982.

[30] P. Wolper, Temporal logic can be more expressive, *Inform. and Control* **56** (1983) 72–99.