Minimal Büchi Automata for Certain Classes of LTL Formulas

Jacek Cichoń*, Adam Czubak†, Andrzej Jasiński†

*Institute of Mathematics and Computer Science
Wrocław University of Technology, Poland,
†Institute of Mathematics and Computer Science
Opole University, Poland

Abstract

In this paper we calculate the minimal number of states of Büchi automata which encode some classes of linear temporal logic (LTL) formulas that are frequently used in model checking. Usually the translation of LTL formulas into Büchi automata is generation of automata with as few states as possible in a reasonable time. Testing Automata, e.t.c.) and a number of translation algorithms has been built (see [2]–[5]). The main goal of these algorithms is modeling the system, is investigated (see e.g. SPIN [1]). Therefore the complexity of the resulting task depends highly on the automata and formal languages. Usually the negation of the LTL formula, which express the desired property of the system, is defined as follows:

\[ (\pi, i) \models \neg \phi \iff (\pi, i) \not\models \phi; \]

\[ (\pi, i) \models \phi \land \psi \iff (\pi, i) \models \phi \land (\pi, i) \models \psi; \]

\[ (\pi, i) \models \neg \phi \iff (\pi, i+1) \models \phi; \]

\[ (\pi, i) \models \phi U \psi \iff (\exists j \geq i)((\pi, j) \models \psi) \land (\forall k)(i \leq k < j \rightarrow (\pi, k) \models \phi). \]

I. Introduction

The theoretic approach to model checking is based on the correspondence between linear temporal logic (LTL), Büchi automata and formal languages. Usually the negation of the LTL formula, which express the desired property of the system, is translated into an equivalent Büchi automaton $A_{\neg \phi}$ and then the product $S \times A_{\neg \phi}$, where $S$ is the automaton used for modeling the system, is investigated (see e.g. SPIN [1]). Therefore the complexity of the resulting task depends highly on the size of the automaton obtained from the translation of the LTL formula. In hope of finding efficient translators many different kinds of automata have been investigated (Büchi Automata, Generalized Büchi Automata, Very Weak Alternating Automata, Testing Automata, e.t.c.) and a number of translation algorithms has been built (see [2]–[5]). The main goal of these algorithms is generation of automata with as few states as possible in a reasonable time.

In this paper we find the minimal number of states of Büchi automata which encode some relatively simple LTL formulas. All of these formulas are in common use in model checking and have been considered in literature (see [6]–[8]). Let us stress that in this paper we investigate only one aspect of the complexity of the Büchi automata, namely the number of states. Other important metrics of complexity, such as the number of transitions, are not considered here.

We observed that even for very simple formulas the corresponding Büchi automata are exponentially large. It follows from Theorem 3.3 that the formula $\Diamond p_1 \land \ldots \land \Diamond p_n$ yields a Büchi automaton of size $2^n$. Similarly, the formula $\Diamond(p_1 \land q_1) \land \ldots \land \Diamond(p_n \land q_n)$ requires an automaton with $3^n$ states (see Theorem 3.6).

In Section 2 we recall some basic notions and facts about linear temporal logic and Büchi automata. In Section 3 we prove our main results and in Section 4 we compare our results with the automata generated by the two currently used translators of LTL formulas into Büchi automata: LTL2BA (see [3]) and SPOT (see [9]).

II. Preliminaries

By $\omega$ we denote the set of all natural numbers. Let $[n] = \{0, \ldots, n\}$, for $n \in \omega$. The power set of a given set $X$ is denoted by $2^X$. The concatenation of two finite sequences is denoted by the symbol $\ast$. If $\sigma = (a_1, \ldots, a_n)$ is a finite sequence then by $\sigma^{rev}$ we denote the reverse of $\sigma$, i.e. $\sigma^{rev} = (a_n, \ldots, a_1)$. By $S_n$ we denote the group of permutations of the set \{1, \ldots, n\}.

Linear Temporal Logic (LTL) is used to specify the properties of a system. The language $L(P)$ of LTL (see [10]) is built from a finite set $P$ of propositional variables with standard logical connectives and temporal operators $\Diamond$ (next), $U$ (until), $\Diamond$ (eventually), $\Box$ (always). An LTL formula can be evaluated over a sequence (computation) $\pi \in (2^P)^\omega$. The relation $(\pi, i) \models \phi$ is defined as follows:

1) $(\pi, i) \models p$ for $p \in P$ iff $p \in \pi(i)$;
2) $(\pi, i) \models \neg \phi$ iff $(\pi, i) \not\models \phi$;
3) $(\pi, i) \models \phi \land \psi$ iff $(\pi, i) \models \phi$ and $(\pi, i) \models \psi$;
4) $(\pi, i) \models \Box \phi$ iff $(\pi, i+1) \models \phi$;
5) $(\pi, i) \models \phi U \psi$ iff $(\exists j \geq i)((\pi, j) \models \psi) \land (\forall k)(i \leq k < j \rightarrow (\pi, k) \models \phi)$;
6) $(\pi, i) \models \Diamond \phi$ iff $(\forall j \geq i)((\pi, j) \models \phi)$;
Finally, the computation $\pi$ satisfies a formula $\phi$, that we denote by $\pi \models \phi$, if $(\pi, 0) \models \phi$. The models of the LTL formula $\phi$ are defined as

$$\mod(\phi) = \{ \pi \in (2^P)^\omega : \pi \models \phi \}.$$ 

A Büchi automaton (see [11]) is a tuple $A = (\Sigma, S, S_0, \rho, F)$, where $\Sigma$ is a finite set called alphabet, $S$ is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $\rho : S \times \Sigma \to 2^S$ is a transition function and $F$ is a set of accepting states. By $|A|$ we denote the number of states of $A$. Elements of the set $\Sigma^\omega$ are called $\omega$-words over the alphabet $\Sigma$. The inputs of $A$ are $\omega$-words over $\Sigma$. A run over the $\omega$-word $w = (a_n)_{n \in \omega}$ is a sequence of states $r = (r_n)_{n \in \omega} \in S^\omega$, such that $r_0 \in S_0$ and $r_{i+1} = \rho(r_i, a_i)$ for each $i \geq 0$. Let

$$\Inf(r) = \{ s \in S : s = r_i \text{ for infinitely many } i \-s \}.$$ 

The Büchi automaton accepts the $\omega$-word $w$ if there exists a run $r$ over $w$ such that $\Inf(r) \cap F \neq \emptyset$. We denote by $L_\omega(A)$ the set of all $\omega$-words accepted by the automaton $A$. The following classical theorem (see [12]) connects LTL formulas with Büchi automata:

**Theorem 2.1:** For each $\phi \in L(P)$ there exists a Büchi automaton $A_\phi$ over the alphabet $2^P$ such that

$$\mod(\phi) = L_\omega(A_\phi).$$

We say that the automaton $A$ encodes the formula $\phi$ if $\mod(\phi) = L_\omega(A)$.

**III. Büchi State Complexity for LTL Formulas**

The complexity of formal model checking method based on LTL depends highly on the size of the automaton obtained from the translation of the LTL formula. The following definition formalizes this notion:

**Definition 3.1:** The Büchi state complexity $\bsc(\phi)$ of the formula $\phi \in L(P)$ is the minimal number of states of a Büchi automaton $A$ which encodes the formula $\phi$.

For example, it is easy to check that $\bsc(\square p) = 1$, $\bsc(\Diamond p) = \bsc(\square \Diamond p) = \bsc(\Diamond \square p) = 2$ and $\bsc(\Diamond(p \land \Diamond q)) = 3$. In the following sections we shall calculate the Büchi state complexity of some LTL formulas commonly used in formal model checking (see e.g. [3]).

**A. Something Will Occur n Times in a Row**

Let $p$ be a fixed propositional variable. Let $\phi_1(p) = p$ and $\phi_n(p) = p \land \Diamond (\phi_{n-1}(p))$ for $n \geq 2$. Finally, for $n > 0$ we put $N_n(p) = \Diamond(\phi_n(p))$.

Observe that $N_1(p) = \Diamond p$, $N_2(p) = \Diamond(p \land \Diamond p)$, $N_3(p) = \Diamond(p \land \Diamond(p \land \Diamond p))$, and so on. The informal interpretation of the formula $N_n(p)$ is „at some point the proposition $p$ will occur $n$ times in a row.”

**Theorem 3.2:** Let $p, q$ be different propositional variables and let $n, m > 0$. Then

$$\bsc(N_n(p) \land N_m(q)) = (n + 1)(m + 1).$$

**Proof:** Let $\Sigma = 2^{\{p,q\}}$. For $r \in \{p, q\}$, $N > 0$, $k \in [N]$ and $X \in \Sigma$ we put

$$\Delta_{N,r}(k, X) = \begin{cases} N & r = k \\
 k + 1 & r \in X \land k < N \\
 0 & r \notin X \land k < N \end{cases}.$$ 

Let $\rho((k_1, k_2), X) = \{ (\Delta_{n,p}(k_1, X), \Delta_{m,q}(k_2, X)) \}$, where $k_1 \in [n]$ and $k_2 \in [m]$. Finally we put $A = (\Sigma, [n] \times [m], \{(0, 0)\}, \rho, \{(n, m)\})$.

It is easy to see that

$$L_\omega(A) = \mod(N_n(p) \land N_m(q)).$$

Therefore $\bsc(N_n(p) \land N_m(q)) \leq (n + 1)(m + 1)$.

Let us assume now that there exists a Büchi automaton $A$ which encodes the formula $N_n(p) \land N_m(q)$ such that $|A| < (n + 1)(m + 1)$. For each pair $(\alpha, \beta) \in [n] \times [m]$ we define the sequence

$$\sigma_{\alpha, \beta} = (X_1, X_2, \ldots, X_\gamma) \in \Sigma^\gamma,$$
where $\gamma = \max(\alpha, \beta)$ and

$$p \in X_i \iff i \leq \alpha, \quad q \in X_i \iff i \leq \beta.$$  

Next we put $\pi_{\alpha, \beta}^{n, m} = (\sigma_{\alpha, \beta})^{rev} * \sigma_{n-\alpha, m-\beta} * \emptyset^\omega$. Notice that, for example,

$$\pi_{4, 2}^{5, 4} = (\{p\}, \{p\}, \{p, q\}, \{p, q\}, \{q\}) * \emptyset^\omega.$$  

It is easy to check that for all pairs $(\alpha, \beta) \in [n] \times [m]$ we have $\pi_{\alpha, \beta}^{n, m} \in \text{mod}(N_n(p) \land N_m(q))$. For all $(\alpha, \beta) \in [n] \times [m]$ we fix runs $r_{\alpha, \beta}$ of the automaton $A$ which accepts $\pi_{\alpha, \beta}^{n, m}$, i.e. a sequence of states

$$r_{\alpha, \beta} = (s_0, s_1, s_2, \ldots, s_\gamma, \ldots),$$  

such that $s_0$ is an initial state, $s_i \in \rho(s_{i-1}, \pi_{\alpha, \beta}^{n, m}(i))$ for $1 \leq i \leq n$ and $\text{Inf}(r_{\alpha, \beta}) \cap F \neq \emptyset$. Let $y_{\alpha, \beta} = r_{\alpha, \beta}(\max(\alpha, \beta))$. From the assumption $|A| < (n+1)(m+1)$ and the pigeonhole principle we deduce that there are two pairs $(\alpha, \beta) \neq (\alpha', \beta')$ such that $y_{\alpha, \beta} = y_{\alpha', \beta'}$. We can assume that $\alpha < \alpha'$. Then

$$\tilde{r} = r_{\alpha, \beta} \upharpoonright (0, \ldots, \max(\alpha, \beta)) * r_{\alpha', \beta'} \upharpoonright (\max(\alpha', \beta') + 1, \ldots, \infty)$$  

is an accepting run of the automaton $\tilde{A}$. Moreover, this is a run over the $\omega$-word

$$(\sigma_{\alpha, \beta})^{rev} * \sigma_{n-\alpha', m-\beta'} * \emptyset^\omega.$$  

But $\alpha + (n-\alpha') = n - (\alpha' - \alpha) < n$, so the automaton $\tilde{A}$ accepts an $\omega$-word in which there are strictly less than $n$ occurrences of the propositional variable $p$, which is impossible. Therefore we have proved that $\text{bsc}(N_n(p) \land N_m(q)) \geq (n+1)(m+1)$.

**Theorem 3.3**: Let $p_1, \ldots, p_k$ be pairwise different propositional variables and let $n_1, \ldots, n_k$ be positive natural numbers. Then

$$\text{bsc}\left(\bigwedge_{i=1}^{k} N_{n_i}(p_i)\right) = \prod_{i=1}^{k} (n_i + 1).$$

**Proof**: The proof for this theorem is a generalization of the proof of Theorem 2. Namely, we consider the automaton with states $[n_1] \times \ldots \times [n_k]$ in the first part of the proof and later we simulate the run on an arbitrary automaton by sequences $\pi_{n_1, \ldots, n_k}^{n, m}$.

**Remark 3.4**: From Theorem 3.3 we get $\text{bsc}(\bigotimes p_1 \land \ldots \land \bigotimes p_k) = 2^k$, $\text{bsc}(\bigotimes (p_1 \land \bigcirc p_1) \land \ldots \land \bigotimes (p_k \land \bigcirc p_k)) = 3^k$, and so on. Therefore we see that even for very simple LTL formulas the size of encoding Büchi automata is exponential in the size of formulas.

**B. Some Sequence will Occur in the Future in the Proper Order**

Let $n > 0$ and let $p_1, \ldots, p_n$ be fixed propositional variables. We define recursively the formula $\phi_n(p_1, \ldots, p_n)$ as follows:

$$\phi_1(p_1, \ldots, p_n) = p_n, \quad \phi_i(p_1, \ldots, p_n) = p_{n-i+1} \land \bigotimes (\phi_{i-1}(p_1, \ldots, p_n)) \text{ for } i = 2, \ldots, n.$$  

Finally, we put

$$E_n(p_1, \ldots, p_n) = \bigotimes (\phi_n(p_1, \ldots, p_n)).$$  

Observe that $E_1(p_1) = \bigotimes p_1$, $E_2(p_1, p_2) = \bigotimes (p_1 \land \bigotimes p_2)$,

$$E_3(p_1, p_2, p_3) = \bigotimes (p_1 \land \bigotimes (p_2 \land \bigotimes p_3))$$  

and so on. The informal interpretation of the formula $E_n(p_1, \ldots, p_n)$ is „the sequence of events $p_1, \ldots, p_n$ will occur in future in the proper order.” We shall prove a theorem similar to Theorem 3.2. In fact not only this theorem but also its proof is similar to the proof of the previous one, but we have found out that the common generalization of both proofs is artificial.

**Theorem 3.5**: Let $n, m > 0$ and let $p_1, \ldots, p_n, q_1, \ldots, q_m$ be pairwise different propositional variables. Then

$$\text{bsc}(E_n(p_1, \ldots, p_n) \land E_m(q_1, \ldots, q_m)) = (n+1)(m+1).$$

**Proof**: Let $\Sigma = 2^{\{p_1, p_2, \ldots, p_n, q_1, q_2, \ldots, q_m\}}$. For $N \in \omega$, $k \in [N]$ and $X \in \Sigma$ we define

$$\Delta_N(k, X) = \begin{cases} N & k = N \\ k + s & p_{k+1}, \ldots, p_{k+s} \in X \land k < N \end{cases}$$  

and

$$\Theta_n(k, X) = \begin{cases} N & k = N \\ k + s & q_{k+1}, \ldots, q_{k+s} \in X \land k < N \end{cases}$$  

Let

$$\rho((k_1, k_2), X) = \{(\Delta_n(k_1, X), \Theta_m(k_2, X))\},$$
where \( k_1 \in [n] \) and \( k_2 \in [m] \). Finally, we put
\[
\mathcal{A} = (\Sigma, [n] \times [m], \{(0, 0)\}, \rho, \{(n, m)\})
\]
It is easy to see that
\[
\mathcal{L}_\omega(\mathcal{A}) = \text{mod}(E_n(p_1, p_2, \ldots, p_n) \land E_m(q_1, q_2, \ldots, q_m)).
\]
Therefore \( \text{bsc}(E_n(p_1, p_2, \ldots, p_n) \land E_m(q_1, q_2, \ldots, q_m)) \leq (n + 1)(m + 1) \).

Let us assume now that there exists a Büchi automaton \( \mathcal{A} \) which encodes the formula \( E_n(p_1, \ldots, p_n) \land E_m(q_1, \ldots, q_m) \) such that \( |\mathcal{A}| < (n + 1)(m + 1) \). For each pair \((\alpha, \beta) \in [n] \times [m]\) we define the sequence
\[
\sigma_{\alpha, \beta} = (X_1, X_2, \ldots, X_\gamma) \in \Sigma^\gamma
\]
where \( \gamma = \max(\alpha, \beta) \) and
\[
p_i \in X_j \leftrightarrow j = \gamma - \alpha + i, \quad q_i \in X_j \leftrightarrow j = \gamma - \beta + i
\]
For the sequence \( \sigma_{\alpha, \beta} \) we define the sequence \( \sigma_{\alpha, \beta}^{\text{end}} = (X_1, X_2, \ldots, X_\delta) \in \Sigma^\delta \) where \( \delta = \max(n - \alpha, m - \beta) \) and
\[
p_{\alpha + i} \in X_i \leftrightarrow i \leq n - \alpha, \quad q_{\beta + i} \in X_i \leftrightarrow i \leq m - \beta.
\]
Next we put \( \pi_{\alpha, \beta}^{n,m} = \sigma_{\alpha, \beta} \ast \sigma_{\alpha, \beta}^{\text{end}} \ast \emptyset^\omega \). Notice that, for example,
\[
\pi_{4,5}^{5,4} = (\{p_1\}, \{p_2\}, \{p_3, q_1\}, \{p_4, q_2\}, \{p_5, q_3\}, \{q_4\}) \ast \emptyset^\omega.
\]
It is easy to observe that \( \pi_{\alpha, \beta}^{n,m} \in \text{mod}(E_n(p_1, p_2, \ldots, p_n) \land E_m(q_1, q_2, \ldots, q_m)) \) for all pairs \((\alpha, \beta) \in [n] \times [m]\). Now, similarly as in the proof of Theorem 3.3, we fix a run \( r_{\alpha, \beta} \) of automaton \( \mathcal{A} \) which accepts \( \pi_{\alpha, \beta}^{n,m} \) and we deduce that there are two pairs \((\alpha, \beta) \neq (\alpha', \beta')\) such that \( r_{\alpha, \beta}(\max(\alpha, \beta)) = r_{\alpha', \beta'}(\max(\alpha', \beta')) \). Finally we deduce that
\[
\bar{r} = (r_{\alpha, \beta}[0, \ldots, \max(\alpha, \beta)]) \ast (r_{\alpha', \beta'}[\max(\alpha', \beta') + 1, \ldots, \infty])
\]
is an accepting run of automaton \( \mathcal{A} \) over the \( \omega \)-word
\[
\sigma_{\alpha, \beta} \ast \sigma_{\alpha, \beta}^{\text{end}} \ast \emptyset^\omega.
\]
We can assume that \( \alpha < \alpha' \). Then the automaton \( \mathcal{A} \) accepts a word in which there are no propositional variables \( p_{\alpha + 1}, p_{\alpha + 2}, \ldots, p_{\alpha'} \), which is impossible. Therefore we have proved that
\[
\text{bsc}(E_n(p_1, \ldots, p_n) \land E_m(q_1, \ldots, q_m)) \geq (n + 1)(m + 1)
\]
Using similar arguments we can prove the following generalization of the previous Theorem:

**Theorem 3.6:** Let \( n_1, \ldots, n_k \) be positive natural numbers and \( \{p_i, j_i: i \in \{1, 2, \ldots, k\}, j_i \in \{1, 2, \ldots, n_i\}\} \) be pairwise different propositional variables. Then
\[
\text{bsc}( \bigwedge_{i=1}^k E_n(p_{i,1}, p_{i,2} \ldots p_{i,n_i}) ) = \prod_{i=1}^k (n_i + 1)
\]

**C. All Events Occur Infinitely Often**

For each positive natural number \( k \) we put
\[
\psi_k = \bigwedge_{i=1}^k (\Box\Diamond p_i).
\]
This kind of formulas often appears in various fairness conditions. The informal interpretation of the formula \( \psi_k \) is „all of the propositions \( p_1, \ldots, p_k \) occur infinitely often in an arbitrary order“.

**Theorem 3.7:** \((\forall k > 0) \left( \text{bsc}(\bigwedge_{i=1}^k (\Box\Diamond p_i)) = k + 1 \right) \)

**Proof:** Let \( \Sigma = 2^{\{p_1, p_2, \ldots, p_k\}} \). For \( X \in \Sigma \) and \( m \in [k] \) we define
\[
\rho(m, X) = \begin{cases} 
0 & : m = k \\
m + 1 & : p_{m+1} \in X \land m < k \\
m & : p_{m+1} \notin X \land m < k
\end{cases}
\]
Finally we put
\[
\mathcal{A} = (\Sigma, [k], \{0\}, \rho, \{k\})
\]
It is easy to check that
\[ \mathcal{L}_\omega(\mathbb{A}) = \mod(\psi_k), \]
therefore \( \text{bsc}(\psi_k) \leq (k + 1). \)

It is easy to check that if \( k \leq 3 \) then \( \text{bsc}(\psi_k) \geq k + 1. \) We shall assume from now on that \( k > 3. \) Now let us suppose \( \mathbb{A} = (\Sigma, S, S_0, \rho, F) \) is a Büchi automaton which encodes the formula \( \psi_k \) and \( |A| \leq k. \) Let us consider the \( \omega \)-word
\[ \bar{w} = (\{p_1\}, \{p_2\}, \ldots, \{p_k\})^\omega. \]

Obviously \( \bar{w} \in \mod(\psi_k). \) We fix an accepting run \( \bar{r} \) of the automaton \( \mathbb{A} \) over the word \( \bar{w}. \) Let \( s \in \text{Inf}(\bar{r}) \) be an accepting state which occurs infinitely often in the run \( \bar{r}. \) Then \( \bar{r} \) can be represented in the following way:
\[ \bar{r} = (q_0, \ldots, q_{r_0}, s, q_1, \ldots, q_{r_1}, s, q_2, \ldots, q_{r_2}, s, \ldots) \]
It is easy to observe that \( s \not\in \rho(s, \{p_i\}) \) for all \( i \in \{1, 2, \ldots, k\}. \) We call a single event run a finite sequence \( (Q_1, Q_2, \ldots, Q_r) \) of the state of the automaton such that
\[ (\forall i \in \{1, 2, \ldots, r - 1\})(\exists j \in \{1, 2, \ldots, k\})(Q_{i+1} \in \rho(Q_i, \{p_j\})). \]

Observe that each segment \( (s, q_{i,1}, q_{i,2}, \ldots, q_{i,r}, s) \) of the run \( \bar{r} \) for \( t \geq 1 \) is a single event run. Take the shortest one and denote it by \( \eta = (\eta_0, \eta_1, \eta_2, \ldots, \eta_r, \eta_{r+1}), \) where \( \eta_0 = \eta_{r+1} = s. \) From the fact that \( \eta \) is the shortest one we deduce that \( (\forall i < j \leq r)(\eta_i \neq \eta_j). \) Let \( \omega_\eta = (\{\alpha_0\}, \ldots, \{\alpha_r\}) \) be the subword of the word \( \bar{w} \) corresponding to the segment \( \eta. \) It is clear that \( \{p_1, p_2, \ldots, p_k\} \subseteq \{\alpha_0, \alpha_1, \ldots, \alpha_r\} \) so from the assumption \( |A| \leq k \) we deduce that \( \{p_1, p_2, \ldots, p_k\} = \{\alpha_0, \alpha_1, \ldots, \alpha_r\}. \) We will show that not all states of the automaton \( \mathbb{A} \) are accepting ones, i.e. that \( |F| < k. \) Namely, suppose that \( F = S \) and consider the sequence \( \delta = ((\{p_1\}, \{p_2\}, \ldots, \{p_k\})^0^\omega. \) The sequence \( \delta \) is accepted by \( \mathbb{A}, \) so there exists a state \( q \in S \) such that \( \rho(q, \emptyset) \neq \emptyset. \) If \( q \in \rho(q, \emptyset) \) then some sequence of the form \( \sigma * 0^\omega \) would be accepted by the automaton \( \mathbb{A}, \) though \( \sigma * 0^\omega \not\in \mod(\psi_k). \) On the other hand if there exists a state \( q_i \neq q \) such that \( q_i \in \rho(q_i, \emptyset), \) then the automaton \( \mathbb{A} \) would accept a sequence of the form \( \sigma * (0 * \lambda)^\omega, \) where not all propositional variables \( p_1, \ldots, p_k \) occur in \( \lambda. \)

Notice that \( \mathcal{L}_\omega(\mathbb{A}) = \bigcup_{f \in F} \mathcal{L}_\omega(\mathbb{A}_f), \) where
\[ \mathbb{A}_f = (\Sigma, S, S_0, \rho, \{f\}). \]

Let us fix a finite state \( \eta_i \in F. \) For each permutation \( \Pi \in S_k \) we define the sequence \( x_\Pi = (\{\alpha_{\Pi(1)}\}, \{\alpha_{\Pi(2)}\}, \ldots, \{\alpha_{\Pi(k)}\})^\omega. \)
It is easy to observe that if \( x_\Pi \in \mathcal{L}_\omega(\mathbb{A}_{\eta_i}) \) then
\[ x_\Pi = \sigma * (\alpha_{(i-1)} \mod k) * (\alpha_i) * (\alpha_{(i+1)} \mod k) * \lambda \]
for some \( \sigma \in \Sigma^* \) and \( \lambda \in \Sigma^\omega. \) Let \( a = \alpha_{(i-1)} \mod k, b = \alpha_i, \) and \( c = \alpha_{(i+1)} \mod k. \)

There are \((k - 2)!\) permutations \( \Pi \in S_k \) such that \( (a, b, c) \) is a subsequence of \( (\{\alpha_{\Pi(1)}\}, \{\alpha_{\Pi(2)}\}, \ldots, \{\alpha_{\Pi(k)}\}). \) There are \((k - 3)!\) permutations \( \Pi \in S_k \) such that \( \alpha_{\Pi(1)} = c, \alpha_{\Pi(k-1)} = a \) and \( \alpha_{\Pi(k)} = b \) and there are \((k - 3)!\) permutations \( \Pi \in S_k \) such that \( \alpha_{\Pi(1)} = b, \alpha_{\Pi(2)} = c \) and \( \alpha_{\Pi(k)} = a. \) Therefore
\[ |\Pi \in S_k : x_\Pi \in \mathcal{L}_\omega(\mathbb{A}_{\eta_i})| \leq (k - 2)! + 2(k - 3)!. \]
Recall that for each \( \Pi \in S_k \) we have \( x_\Pi \in \mathcal{L}_\omega(\mathbb{A}). \) Therefore
\[ k! \leq |F|((k - 2)! + 2(k - 3)!). \]

But we showed that \( |F| \leq k - 1, \) so
\[ k! \leq (k - 1)((k - 2)! + 2(k - 3)!), \]
which is not true for \( k > 3. \) Therefore we have proved that \( \text{bsc}(\psi_k) \geq k + 1. \)

**D. One of the Events Eventually Holds Forever**

In this section we take under consideration the negation of the formula from the previous section, namely, for each positive natural number \( k \) we put
\[ \xi_k = \bigvee_{i=1}^{k} (\Diamond p_i). \]

The informal interpretation of the formula \( \xi_k \) is „one of the propositions \( p_1, \ldots, p_k \) eventually holds forever.”

**Theorem 3.8:** \((\forall k > 0) \left( \text{bsc}(\bigvee_{i=1}^{k} (\Diamond p_i)) = k + 1 \right)\)
Proof: Let $\Sigma = 2^{\{p_1, p_2, \ldots, p_k\}}$. For $X \in \Sigma$ we put $\rho(0, X) = \{0\} \cup \{i : p_i \in X\}$ and

$$\rho(i, X) = \begin{cases} \{i\} & : p_i \in X, \\ \emptyset & : p_i \notin X \end{cases}$$

for $i \in \{1, \ldots, k\}$. Finally we put

$$A = (\Sigma, [k], \{0\}, \rho, \{1, 2, \ldots, k\}).$$

It is easy to check that $L_\omega(A) = \text{mod}(\xi_k)$, so $\text{bsc}(\xi_k) \leq (k + 1)$.

Suppose that $A$ is a B"uchi automaton such that $L_\omega(A) = \text{mod}(\xi_k)$ and $|A| \leq k$. For $i \in \{1, 2, \ldots, k\}$ we define the sequence $\pi_i = (\{p_j\}^\omega)$. Then $\pi_i \in L_\omega(A)$, so there is a run $r_i$ over the word $\pi_i$ and an accepting state $f_i$ such that $f_i \in \text{Inf}(r_i)$. We show now that $f_i \neq f_j$ for all $i \neq j$. So let us suppose that $i \neq j$ and $f_i = f_j$. Fix $\alpha_i < \beta_i$ and $\alpha_j < \beta_j$ such that

$$r_i(\alpha_i) = r_i(\beta_i) = r_j(\alpha_j) = r_j(\beta_j) = f.$$  

Then the run

$$r_i \upharpoonright (1, \ldots, \alpha_i) * (r_i \upharpoonright (\alpha_i + 1, \ldots, \beta_i) * r_j \upharpoonright (\alpha_j + 1, \ldots, \beta_j))\omega$$

of the automaton $A$ would accept the word

$$\{p_i\}^{\alpha_i} * (\{p_i\}^{\beta_i - \alpha_i} * \{p_j\}^{\beta_j - \alpha_j})\omega,$$

which doesn’t belong to $\text{mod}(\xi_k)$. Therefore we see that all states of $A$ are accepting ones. Let us consider the run $r$ of the automaton $A$ over the $\omega$-word

$$w = \emptyset^k * \{p_i\}^\omega.$$

From the assumption $|A| \leq k$ and the fact that for each state $q$ of the automaton $A$ we have $q \notin \rho(q, \emptyset)$ we deduce that there are $0 \leq i < j \leq k$ such that $r_i = r_j$. But

$$(r_0, r_1, \ldots, r_{i-1}) * (r_i, r_{i+1}, \ldots, r_j)\omega$$

is an accepting run, therefore $\emptyset^\omega \in L_\omega(A)$, which is impossible. 

\section*{IV. Comparison with LTL2BA and SPOT}

We compared our results with two currently used LTL to B"uchi automata translators: LTL2BA\textsuperscript{1} and SPOT\textsuperscript{2}. We compared the number of states generated by these tools and our results from the previous section. Moreover, we checked the time consumed by them. Similar research was done by other authors (see e.g. [8]), but their experiments compared only the relative efficiency of translators—we compared the result of translation with our lower theoretical bounds.

We run all tests on HP Proliant DL360, with 2 Intel(R) Xeon(TM) CPU 5160 Processor (3.00 GHz, 1333 FSB), 2GB of memory. The operating system was FreeBSD 6.2-RELEASE with SMP support. Both SPOT and LTL2BA were compiled on this machine to achieve best performance. For the purpose of tests both programs were configured with the formula simplification enabled.

We summarize the results in the series of tables. In the third and fifth column of each table we have the number of states of the B"uchi automaton generated by LTL2BA and SPOT and in the fourth and sixth column we have the time consumed by these programs to translate the given formula. If the program returned no answer within 24h we marked it with N/A and if a program died then we marked it by RIP.

First we consider the formula $\alpha_n = E_n(p_1, p_2 \ldots p_n) \land E_n(q_1, q_2 \ldots q_n)$, i.e.

$$\alpha_n = \Diamond (p_1 \land \Diamond (p_2 \land \ldots \land \Diamond p_n) \ldots) \land \Diamond (q_1 \land \Diamond (q_2 \land \ldots \land \Diamond q_n) \ldots) .$$

From Theorem 3.6 we have $\text{bsc}(\alpha_n) = (n + 1)^2$. Table I contains the obtained results. We see that the automata generated by both tools are far from being optimal. For example, for $\alpha_2$ the minimal B"uchi automaton has 9 states, but the automata generated by LTL2BA and SPOT have respectively 12 and 15 states. The difference between the optimal automaton and the generated ones grows when $n$ increases.

We must stress that both programs LTL2BA and SPOT produce optimal B"uchi automata for the first class of formulas considered in this paper, namely for formulas of the form $\beta_n = N_n(p) \land N_n(q)$, i.e.

$$\beta_n = \Diamond (p \land \Diamond (p \land \ldots \land \Diamond p) \ldots) \land \Diamond (q \land \Diamond (q \land \ldots \land \Diamond q) \ldots) .$$

\footnotetext[1]{http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/index.php}

\footnotetext[2]{http://spot.lip6.fr/wiki/}
Both tools produced optimal Büchi automata in a very short time. However, the consumed time grows very quickly and SPOT died for formulas that we used in verification of properties of some distributed protocols. Observation from the last section shows that simplifications in the preprocessing stage play a very important role. Notice that the SPOT translator can do such optimizations and they are doing it in a quick time. The results of these experiments are in the Table II. However, let us consider the formula

\[ \beta_n' = \Diamond (p \land \Box p \land \Box^2 p \land \ldots \land \Box^{n-1} p) \land \Diamond (q \land \Box q \land \Box^2 q \land \ldots \land \Box^{n-1} q) \]

The formulas \( \beta_n \) and \( \beta_n' \) are logically equivalent. Table III contains the results of experiments for the formulas \( \beta_n' \). We observed that LTL2BA produces the optimal automaton, but the time requirements increase dramatically. It turns out that the formula simplifications in the preprocessing stage play a very important role. Notice that the SPOT translator can do such optimizations automatically.

Both considered tools produce optimal automata for the formula

\[ \psi_n = \Box \Diamond p_1 \land \Box \Diamond p_2 \land \ldots \land \Box \Diamond p_n \]

however, the consumed time grows very quickly and SPOT died for formulas \( \psi_{19} \) and \( \psi_{20} \). Table IV contains the results for formulas \( \psi_n \). Finally, Table V contains the results of experiments for formulas of the form

\[ \xi_n = \Diamond \Box p_1 \lor \Diamond \Box p_2 \lor \ldots \lor \Diamond \Box p_n \]

Both tools produced optimal Büchi automata in a very short time.

## V. Conclusions and Future Work

All of the LTL formulas analyzed in this paper are widely used in formal verification. In fact we focused in this paper on formulas which we used in verification of properties of some distributed protocols. Observation from the last section shows that the considered translators are far from being ideal. In many simple cases the size of the automata generated by both analyzed translators is bigger than the lower bound (i.e. its Büchi state complexity) or the time consumed by these tools is very big.
In fact we have discussed in this paper some examples of LTL formula patterns (or templates). In formal verification the class of useful LTL-patterns, up to our knowledge, is not too large; it consists of few hundred of templates. Let us call the LTL-patterns recognized if we know the size and the construction of their minimal Büchi automata for each instance of the pattern. Suppose that after some period of investigations most of useful patterns in formal verification will be recognized. Then we would be able to construct a database with these patterns and algorithms for generation of their Büchi automata. This database, when completed, could be used in tools like SPIN for building Büchi automaton and in many (or even most) cases the time for this operation could be reduced from several hours to few seconds needed for the access to the database. Notice also that only one database of this kind is required in the world.

We plan to expand the class of LTL-patterns frequently used in formal verification with precisely calculated Büchi state complexity and to extend our investigations onto other metrics of complexity of Büchi automaton.

ACKNOWLEDGMENT

The paper was partially supported by grant no 342162 of the Institute of Mathematics and Computer Science of Wrocław University of Technology.

REFERENCES