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. Among others, we show that the minimal size of a Büchi automaton accepting the formula $\Box\Diamond p_1 \wedge \ldots \wedge \Box\Diamond p_n$ is $n+1$, the minimal size of Büchi automaton accepting the formula $\Diamond p_1 \wedge \ldots \wedge \Diamond p_n$ is 2^n and the minimal size of a Büchi automaton accepting the formula $\Diamond(p_1 \wedge \Diamond p_1) \wedge \ldots \wedge \Diamond(p_n \wedge \Diamond p_n)$ is 3^n. Our results may be used for verification of the quality of algorithms which automatically translate LTL formulas into Büchi automata and for improving the quality and speed of such translators. In the last section of this paper we compare our lower-bound estimations to Büchi automata generated by two currently used translators: LTL2BA and SPOT. We have checked, among others, that the LTL2BA translator generates a Büchi automaton with 25 states and the SPOT translator generates an automaton with 31 states for the formula $\Diamond(p1 \wedge \Diamond(p2 \wedge \Diamond p3)) \wedge \Diamond(q1 \wedge \Diamond(q2 \wedge \Diamond q3))$, while the minimal required number of states is 16.

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 φ formula, which express the desired property of the system, is translated into an equivalent Büchi automaton $\mathbb{A}_{\neg\varphi}$ and then the product $\mathbb{S} \times \mathbb{A}_{\neg\varphi}$, where $\mathbb{S}$ 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 encodes 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 $\Diamond p_1 \wedge \ldots \wedge \Diamond p_n$ yields a Büchi automaton of size 2^n. Similarly, the formula $\Diamond(p_1 \wedge \Diamond q_1) \wedge \ldots \wedge \Diamond(p_n \wedge \Diamond q_n)$ 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 ω 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 $*$. If $\sigma = (a_1, \ldots, a_n)$ is a finite sequence then by σ^{rev} we denote the reverse of σ, 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 $\mathcal{L}(\mathcal{P})$ of LTL (see [10]) is built from a finite set $\mathcal{P}$ of propositional variables with standard logical connectives and temporal operators $\bigcirc$ (next), U (until), $\Diamond$ (eventually), $\Box$ (always). An LTL formula can be evaluated over a sequence (*computation*) $\pi \in (2^{\mathcal{P}})^\omega$. The relation $(\pi, i) \models \phi$ is defined as follows:

1) $(\pi, i) \models p$ for $p \in \mathcal{P}$ iff $p \in \pi(i)$;
2) $(\pi, i) \models \neg\phi$ iff $(\pi, i) \not\models \phi$;
3) $(\pi, i) \models \phi \wedge \psi$ iff $(\pi, i) \models \phi \wedge (\pi, i) \models \psi$;
4) $(\pi, i) \models \bigcirc\phi$ iff $(\pi, i+1) \models \phi$;
5) $(\pi, i) \models \Diamond\phi$ iff $(\exists j \geq i)((\pi, j) \models \phi)$;
6) $(\pi, i) \models \Box\phi$ iff $(\forall j \geq i)((\pi, j) \models \phi)$;
7) $(\pi, i) \models \phi U \psi$ iff $(\exists j \geq i)((\pi, j) \models \psi) \wedge (\forall k)(i \leq k < j \rightarrow (\pi, k) \models \phi)$.

Finally, the computation π *satisfies* a formula ϕ, that we denote by $\pi \models \phi$, if $(\pi, 0) \models \phi$. The models of the LTL formula ϕ are defined as

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

A Büchi automaton (see [11]) is a tuple $\mathbb{A} = (\Sigma, S, S_0, \rho, F)$, where Σ 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 $|\mathbb{A}|$ we denote the number of states of $\mathbb{A}$. Elements of the set Σ^{ω} are called ω–words over the alphabet Σ. The inputs of $\mathbb{A}$ are ω–words over Σ. A *run* over the ω–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 $r_{i+1} \in \rho(r_i, a_i)$ for each $i \geq 0$. Let

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

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

Theorem 2.1: For each $\phi \in \mathcal{L}(\mathcal{P})$ there exists a Büchi automaton $\mathbb{A}_{\phi}$ over the alphabet $2^{\mathcal{P}}$ such that

$$\mathrm{mod}(\phi) = \mathcal{L}_{\omega}(\mathbb{A}_{\phi}) \ .$$

We say that the automaton $\mathbb{A}$ *encodes* the formula ϕ if $\mathrm{mod}(\phi) = \mathcal{L}_{\omega}(\mathbb{A})$.

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 $\mathbf{bsc}(\phi)$ of the formula $\phi \in \mathcal{L}(\mathcal{P})$ is the minimal number of states of a Büchi automaton $\mathbb{A}$ which encodes the formula ϕ.

For example, it is easy to check that $\mathbf{bsc}(\Box p) = 1$, $\mathbf{bsc}(\Diamond p) = \mathbf{bsc}(\Box\Diamond p) = \mathbf{bsc}(\Diamond\Box p) = 2$ and $\mathbf{bsc}(\Diamond(p \wedge \Diamond q)) = 3$. 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 $\phi_n(p) = p \wedge \bigcirc(\phi_{n-1}(p))$ for $n \geq 2$. Finally, for $n > 0$ we put

$$N_n(p) = \Diamond(\phi_n(p)) \ .$$

Observe that $N_1(p) = \Diamond p$, $N_2(p) = \Diamond(p \wedge \bigcirc p)$, $N_3(p) = \Diamond(p \wedge \bigcirc(p \wedge \bigcirc p))$, 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

$$\mathbf{bsc}(N_n(p) \wedge N_m(q)) = (n+1)(m+1) \ .$$

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

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

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

$$\mathbb{A} = (\Sigma, [n] \times [m], \{(0,0)\}, \rho, \{(n,m)\}) \ .$$

It is easy to see that

$$\mathcal{L}_{\omega}(\mathbb{A}) = \mathrm{mod}(N_n(p) \wedge N_m(q)) \ .$$

Therefore $\mathbf{bsc}(N_n(p) \wedge N_m(q)) \leq (n+1)(m+1)$.

Let us assume now that there exists a Büchi automaton $\mathbb{A}$ which encodes the formula $N_n(p) \wedge N_m(q)$ such that $|\mathbb{A}| < (n+1)(m+1)$. 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

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

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

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

It is easy to check that for all pairs $(\alpha,\beta) \in [n] \times [m]$ we have $\pi^{n,m}_{\alpha,\beta} \in \mathrm{mod}(N_n(p) \wedge N_m(q))$. For all $(\alpha,\beta) \in [n] \times [m]$ we fix runs $r_{\alpha,\beta}$ of the automaton $\mathbb{A}$ which accepts $\pi^{n,m}_{\alpha,\beta}$, 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, $s_i \in \rho(s_{i-1}, \pi^{n,m}_{\alpha,\beta}(i))$ for $1 \leq i$ and $Inf(r_{\alpha,\beta}) \cap F \neq \emptyset$. Let $y_{\alpha,\beta} = r_{\alpha,\beta}(\max(\alpha,\beta))$. From the assumption $|\mathbb{A}| < (n+1)(m+1)$ 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

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

is an accepting run of the automaton $\mathbb{A}$. Moreover, this is a run over the ω–word

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

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

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

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

Proof: The proof for this theorem is a generalization of the proof of Theorem 2. Namely, we consider the automaton with states $[n_1] \times \ldots [n_k]$ in the first part of the proof and later we simulate the run on an arbitrary automaton by sequences $\pi^{n_1,\ldots,n_k}_{\alpha_1,\ldots,\alpha_k}$. ■

Remark 3.4: From Theorem 3.3 we get **bsc**$(\Diamond p_1 \wedge \ldots \wedge \Diamond p_k) = 2^k$, **bsc**$(\Diamond(p_1 \wedge \bigcirc p_1) \wedge \ldots \wedge \Diamond(p_k \wedge \bigcirc p_k)) = 3^k$, 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: $\phi_1(p_1,\ldots,p_n) = p_n$, $\phi_i(p_1,\ldots,p_n) = p_{n-i+1} \ \wedge \ \Diamond \ (\phi_{i-1}(p_1,\ldots,p_n))$ for $i = 2\ldots,n$. Finally, we put

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

Observe that $E_1(p_1) = \Diamond p_1$, $E_2(p_1,p_2) = \Diamond(p_1 \wedge \Diamond p_2)$,

$$E_3(p_1,p_2,p_3) = \Diamond(p_1 \wedge \Diamond(p_2 \wedge \Diamond p_3))$$

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

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

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

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

and

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

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

$$\mathbb{A} = (\Sigma, [n] \times [m], \{(0,0)\}, \rho, \{(n,m)\}) \ \ .$$

It is easy to see that

$$\mathcal{L}_\omega(\mathbb{A}) = \mathrm{mod}(E_n(p_1, p_2, \ldots, p_n) \wedge E_m(q_1, q_2, \ldots, q_m)).$$

Therefore **bsc**$(E_n(p_1, p_2, \ldots, p_n) \wedge E_m(q_1, q_2 \ldots, q_m)) \leq (n+1)(m+1)$.

Let us assume now that there exists a Büchi automaton $\mathbb{A}$ which encodes the formula $E_n(p_1, \ldots, p_n) \wedge E_m(q_1, \ldots, q_m)$ such that $|\mathbb{A}| < (n+1)(m+1)$. 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 $\sigma^{end}_{\alpha,\beta} = (X_1, X_2, \ldots, X_\delta) \in \Sigma^\delta$ 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 $\pi^{n,m}_{\alpha,\beta} = \sigma_{\alpha,\beta} * \sigma^{end}_{\alpha,\beta} * \emptyset^\omega$. Notice that, for example,

$$\pi^{5,4}_{4,2} = (\{p_1\}, \{p_2\}, \{p_3, q_1\}, \{p_4, q_2\}, \{p_5, q_3\}, \{q_4\}) * \emptyset^\omega \ \ .$$

It is easy to observe that $\pi^{n,m}_{\alpha,\beta} \in \mathrm{mod}(E_n(p_1, p_2, \ldots, p_n) \wedge E_m(q_1, q_2, \ldots, q_m))$ 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 $\mathbb{A}$ which accepts $\pi^{n,m}_{\alpha,\beta}$ 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

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

is an accepting run of automaton $\mathbb{A}$ over the ω–word

$$\sigma_{\alpha,\beta} * \sigma^{end}_{\alpha',\beta'} * \emptyset^\omega \ \ .$$

We can assume that $\alpha < \alpha'$. Then the automaton $\mathbb{A}$ 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

$$\textbf{bsc}(E_n(p_1, \ldots, p_n) \wedge E_m(q_1, \ldots, q_m)) \geq (n+1)(m+1) \ \ .$$

■

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

$$\textbf{bsc}(\bigwedge_{i=1}^{k} E_{n_i}(p_{i,1}, p_{i,2} \ldots p_{i,n_i})) = \prod_{i=1}^{k}(n_i + 1) \ \ .$$

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 ψ_k is *„all of the propositions $p_1, \ldots, p_k$ occur infinitely often in an arbitrary order*".

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

Proof: Let $\Sigma = 2^{\{p_1, p_2, \ldots p_k\}}$. For $X \in \Sigma$ and $m \in [k]$ we define

$$\rho(m, X) = \begin{cases} 0 & : \ m = k \\ m+1 & : \ p_{m+1} \in X \wedge m < k \\ m & : \ p_{m+1} \notin X \wedge m < k \end{cases}$$

Finally we put

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

It is easy to check that

$$\mathcal{L}_\omega(\mathbb{A}) = \mathrm{mod}(\psi_k) \ ,$$

therefore $\mathbf{bsc}(\psi_k) \leq (k+1)$.

It is easy to check that if $k \leq 3$ then $\mathbf{bsc}(\psi_k) \geq k+1$. 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 ψ_k and $|\mathbb{A}| \leq k$. Let us consider the ω-word

$$\widetilde{w} = (\{p_1\}, \{p_2\}, \ldots, \{p_k\})^\omega.$$

Obviously $\widetilde{w} \in \mathrm{mod}(\psi_k)$. We fix an accepting run $\widetilde{r}$ of the automaton $\mathbb{A}$ over the word $\widetilde{w}$. Let $s \in Inf(\widetilde{r})$ be an accepting state which occurs infinitely often in the run $\widetilde{r}$. Then $\widetilde{r}$ can be represented in the following way:

$$\widetilde{r} = (q_{0,1}, \ldots, q_{0,r_0}, s, q_{1,1}, \ldots, q_{1,r_1}, s, q_{2,1}, \ldots, q_{2,r_2}, s, \ldots)$$

It is easy to observe that $s \notin \rho(s, \{p_i\})$ 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 $(s, q_{t,1}, q_{t,2}, \ldots, q_{t,r_t}, s)$ of the run $\widetilde{r}$, 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 η is the shortest one we deduce that $(\forall i < j \leq r)(\eta_i \neq \eta_j)$. Let $\widetilde{w_\eta} = (\{\alpha_0\}, \ldots, \{\alpha_r\})$ be the subword of the word $\widetilde{w}$ corresponding to the segment η. It is clear that $\{p_1, p_2, \ldots, p_k\} \subseteq \{\alpha_0, \alpha_1, \ldots, \alpha_r\}$, so from the assumption $|\mathbb{A}| \leq k$ 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 $\delta = (\{p_1\}, \{p_2\}, \ldots, \{p_k\}, \emptyset)^\omega$. The sequence δ 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 $\sigma * \emptyset^\omega$ would be accepted by the automaton $\mathbb{A}$, though $\sigma * \emptyset^\omega \notin \mathrm{mod}(\psi_k)$. On the other hand if there exists a state $q_i \neq q$ such that $q_i \in \rho(q, \emptyset)$, then the automaton $\mathbb{A}$ would accept a sequence of the form $\sigma * (\emptyset * \lambda)^\omega$, where not all propositional variables $p_1, \ldots, p_k$ occur in λ.

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 final 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 $\mathbf{bsc}(\psi_k) \geq k+1$. ■

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

$$\xi_k = \bigvee_{i=1}^{k} (\Diamond \Box p_i) \ .$$

The informal interpretation of the formula ξ_k is „*one of the propositions* $p_1, \ldots, p_k$ *eventually holds forever*."

Theorem 3.8: $(\forall k > 0) \left(\mathbf{bsc}(\bigvee_{i=1}^{k} (\Diamond \Box p_i)) = k+1 \right)$

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

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

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

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

It is easy to check that $\mathcal{L}_\omega(\mathbb{A}) = \mathrm{mod}(\xi_k)$, so **bsc**$(\xi_k) \le (k+1)$.

Suppose that $\mathbb{A}$ is a Büchi automaton such that $\mathcal{L}_\omega(\mathbb{A}) = \mathrm{mod}(\xi_k)$ and $|\mathbb{A}| \le k$. For $i \in \{1, 2, \ldots, k\}$ we define the sequence $\pi_i = (\{p_i\}^\omega)$. Then $\pi_i \in \mathcal{L}_\omega(\mathbb{A})$, so there is a run r_i over the word π_i and an accepting state f_i such that $f_i \in Inf(r_i)$. We show now that $f_i \ne f_j$ for all $i \ne j$. So let us suppose that $i \ne j$ and $f_i = f_j$. Fix $\alpha_i < \beta_i$ and $\alpha_j < \beta_j$ such that

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

Then the run

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

of the automaton $\mathbb{A}$ would accept the word

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

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

$$w = \emptyset^k * \{p_1\}^\omega.$$

From the assumption $|\mathbb{A}| \le k$ and the fact that for each state q of the automaton $\mathbb{A}$ we have $q \notin \rho(q, \{\emptyset\})$ we deduce that there are $0 \le i < j \le k$ 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 $\emptyset^\omega \in \mathcal{L}_\omega(\mathbb{A})$, which is impossible. ■

IV. Comparison with LTL2BA and SPOT

We compared our results with two currently used LTL to Büchi automata translators: LTL2BA[1] and SPOT[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 Büchi 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 $\alpha_n = E_n(p_1, p_2 \ldots p_n) \wedge E_n(q_1, q_2 \ldots q_n)$, i.e.

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

From Theorem 3.6 we have **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 α_2 the minimal Büchi 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 Büchi automata for the first class of formulas considered in this paper, namely for formulas of the form $\beta_n = N_n(p) \wedge N_n(q)$, i.e.

$$\beta_n = \Diamond(p \wedge \bigcirc(p \wedge \ldots \wedge \bigcirc p) \ldots) \wedge \Diamond(q \wedge \bigcirc(q \wedge \ldots \wedge \bigcirc q) \ldots)$$

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

[2]http://spot.lip6.fr/wiki/

TABLE I

FORMULA

$\Diamond(p_1 \wedge \Diamond(p_2 \wedge ... \wedge \Diamond p_n)...) \wedge \Diamond(q_1 \wedge \Diamond(q_2 \wedge \ldots \wedge \Diamond q_n)...)$

		LTL2BA		SPOT	
n	**bsc**(α_n)	States	Time [s]	States	Time [s]
1	4	4	0.01	4	0.073
2	9	12	0.01	15	0.084
3	16	25	0.01	31	0.124
4	25	44	0.01	53	0.236
5	36	69	0.01	81	0.510
6	49	100	0.03	115	1.095
7	64	137	0.05	155	2.248
8	81	180	0.1	201	4.307
9	100	229	0.17	253	7.851
10	121	284	0.29	311	13.719
11	144	345	0.5	375	23.022
12	169	412	0.83	445	36.876
13	196	485	1.32	521	57.716
14	225	564	2.06	603	87.587
15	256	649	3.16	691	129.990
16	289	740	4.81	785	189.464
17	324	837	7.09	887	267.893
18	361	940	10.33	996	374.519
19	400	1049	14.89	1111	517.417
20	441	1164	21.09	1237	701.669

TABLE II

FORMULA

$\Diamond(p \wedge \bigcirc(p \wedge ... \wedge \bigcirc p)...) \wedge \Diamond(q \wedge \bigcirc(q \wedge ... \wedge \bigcirc q)...)$

		LTL2BA		SPOT	
n	**bsc**(β_n)	States	Time [s]	States	Time [s]
1	4	4	0.01	4	0.074
2	9	9	0.01	9	0.077
3	16	16	0.01	16	0.083
4	25	25	0.01	25	0.091
5	36	36	0.01	36	0.101
6	49	49	0.01	49	0.121
7	64	64	0.01	64	0.153
8	81	81	0.01	81	0.197
9	100	100	0.01	100	0.258
10	121	121	0.01	121	0.325
11	144	RIP	RIP	144	0.479
12	169	169	0.02	169	0.651
13	196	RIP	RIP	196	0.882
14	225	225	0.03	225	1.180
15	256	256	0.04	256	1.559
16	289	289	0.05	289	2.039
17	324	324	0.07	324	2.638
18	361	361	0.09	361	3.373
19	400	400	0.11	400	4.275
20	441	441	0.13	441	5.360

TABLE III

FORMULA $\beta'_n = \Diamond(p \wedge \bigcirc p \wedge \bigcirc^2 p \wedge \ldots \wedge \bigcirc^{n-1} p) \wedge \Diamond(q \wedge \bigcirc q \wedge \bigcirc^2 q \wedge \ldots \wedge \bigcirc^{n-1} q)$

		LTL2BA		SPOT	
n	**bsc**(β'_n)	States	Time [s]	States	Time [s]
1	4	4	0.01	4	0.074
2	9	9	0.01	9	0.078
3	16	16	0.01	16	0.081
4	25	25	0.01	25	0.089
5	36	36	0.01	36	0.101
6	49	49	0.01	49	0.119
7	64	64	0.01	64	0.149
8	81	81	0.03	81	0.195
9	100	100	0.06	100	0.261
10	121	121	0.24	121	0.353
11	144	144	0.90	144	0.480
12	169	169	3.62	169	0.655
13	196	196	14.64	196	0.886
14	225	225	61.44	225	1.186
15	256	256	287	256	1.561
16	289	289	1165	289	2.048
17	324	324	4759	324	2.640
18	361	361	19408	361	3.389
19	400	400	53450	400	4.305
20	441	N/A	N/A	441	5.379

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

$$\beta'_n = \Diamond(p \wedge \bigcirc p \wedge \bigcirc^2 p \wedge \ldots \wedge \bigcirc^{n-1} p) \wedge \Diamond(q \wedge \bigcirc q \wedge \bigcirc^2 q \wedge \ldots \wedge \bigcirc^{n-1} q) \ .$$

The formulas β_n and β'_n are logically equivalent. Table III contains the results of experiments for the formulas β'_n. We observed that LTL2BA produces the optimal automatons, 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

$$\psi_n = \Box\Diamond p_1 \wedge \Box\Diamond p_2 \wedge \ldots \wedge \Box\Diamond p_n \ ,$$

however, the consumed time grows very quickly and SPOT died for formulas ψ_{19} and ψ_{20}. Table IV contains the results for formulas ψ_n. Finally, Table V contains the results of experiments for formulas of the form

$$\xi_n = \Diamond\Box p_1 \vee \Diamond\Box p_2 \vee \ldots \vee \Diamond\Box p_n \ .$$

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.

TABLE IV

FORMULA $\psi_n = \Box\Diamond p_1 \wedge \Box\Diamond p_2 \wedge \ldots \wedge \Box\Diamond p_n$

		LTL2BA		SPOT	
n	**bsc**(ψ_n)	States	Time [s]	States	Time [s]
1	2	2	0.01	2	0.074
2	3	3	0.01	3	0.075
3	4	4	0.01	4	0.078
4	5	5	0.01	5	0.085
5	6	6	0.02	6	0.099
6	7	7	0.16	7	0.133
7	8	8	1.30	8	0.212
8	9	9	12.89	9	0.397
9	10	10	106	10	0.828
10	11	11	1209	11	1.833
11	12	12	11442	12	4.149
12	13	N/A	N/A	13	9.415
13	14	N/A	N/A	14	21.296
14	15	N/A	N/A	15	48.123
15	16	N/A	N/A	16	108.402
16	17	N/A	N/A	17	242.422
17	18	N/A	N/A	18	542.459
18	19	N/A	N/A	19	1209.573
19	20	N/A	N/A	RIP	RIP
20	21	N/A	N/A	RIP	RIP

TABLE V

FORMULA $\xi_n = \Diamond\Box p_1 \vee \Diamond\Box p_2 \vee \ldots \vee \Diamond\Box p_n$

		LTL2BA		SPOT	
n	**bsc**(ξ_n)	States	Time [s]	States	Time [s]
1	2	2	0.002	2	0.075
2	3	3	0.002	3	0.076
3	4	4	0.002	4	0.076
4	5	5	0.002	5	0.078
5	6	6	0.002	6	0.079
6	7	7	0.002	7	0.079
7	8	8	0.002	8	0.080
8	9	9	0.002	9	0.081
9	10	10	0.002	10	0.082
10	11	11	0.002	11	0.080
11	12	12	0.002	12	0.081
12	13	13	0.002	13	0.080
13	14	14	0.002	14	0.080
14	15	15	0.002	15	0.082
15	16	16	0.002	16	0.084
16	17	17	0.002	17	0.085
17	18	18	0.002	18	0.086
18	19	19	0.002	19	0.087
19	20	20	0.002	20	0.089
20	21	21	0.002	21	0.090

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 automatons 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 automatons.

ACKNOWLEDGMENT

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

REFERENCES

[1] G. J. Holtzmann, *The Spin Model Checker*. Addison-Wesley, 2003.

[2] R. Gerth, D. Peled, M. Y. Vard, and P. Wolper, "Simple on-the-fly automatic verification of linear temporal logic," in *Proceedings of the 15th IFIP Symp. of Protocol Specification, Testing, and Verification*. North-Holland, 1995, pp. 3–18.

[3] P. Gastin and D.Oddoux, "Fast ltl to büchi automata translation," in *Proceedings of the 13th International Conference on Computer Aided Verification*. Springer-Verlag, 2001, pp. 53–65.

[4] S. Schwon and J. Esperza, *Proceedings of 11th Internat. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)*, ser. LNCS. Springer-Verlag, 2005, vol. 3440, ch. A note on on-the-fly verification algorithms, pp. 174–190.

[5] J. Geldenhuys and A. Valmari, *Proceedings of 10th Internat. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, ser. LNCS. Springer-Verlag, 2005, vol. 2988, ch. Tarjan's algorithm makes on-the-fly LTL verification more efficient, pp. 205–219.

[6] K. Etessami and G. J. Holzmann, *Proceedings of the 11th International Conference of Concurrency Theory*, ser. LNCS. Springer-Verlag, August 2000, vol. 1877, ch. Optimizing Büchi automata, pp. 154–167.

[7] J. Geldenhuys and H. Hansen, *Model Checking Software, 13th Int'l SPIN Workshop*, ser. LNCS. Springer-Verlag, 2006, vol. 3925, ch. Larger automata and less work for LTL model checking, pp. 53–70.

[8] K. Rozier and M. Vardi, "LTL satisfiability checking," in *14th Workshop on Model Checking Software (SPIN '07)*, ser. Lecture Notes in Computer Science (LNCS), vol. 4595. Springer-Verlag, 2007, pp. 149–167.

[9] A. Duret-Lutz and D. Poitrenaud, "Spot: an extensible model checking library using transition-based generalized büchi automata," in *Proceedings of the 12th IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'04)*. IEEE Computer Society Press, 2004, pp. 76–83.

[10] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS 1977)*. IEEE Computer Society Press, 1977, pp. 46–57.

[11] J. Büchi, "On a decision method in restricted second-order arithmetic," in *Proccedings of International Congres on Logic Method and Philosophy of Science, 1960*. Stanford University Press, 1962, pp. 1–12.

[12] P. Wolper, M. Vardi, and A. Sistla, "Reasoning about infinite computation paths," in *Proceedings of the 24th Symposium on Foundations of Computer Science*, 1983, pp. 185–194.