LTL with past and two-way very-weak alternating automata

Paul Gastin and Denis Oddoux

LIAFA, Université Paris 7, 2, place Jussieu, F-75251 Paris Cedex 05
{Paul.Gastin,Denis.Oddoux}@liafa.jussieu.fr

Abstract. In this paper, we propose a translation procedure of PLTL (LTL with past modalities) formulas to Büchi automata using two-way very-weak alternating automata (2VWAA) as an intermiediary step. Our main result is an efficient translation of 2VWAA to generalized Büchi automata (GBA).

1 Introduction

Nowadays, computer systems (both hardware and software) play a central role in most human activities, including safety critical areas. It is therefore essential to improve their reliability. For this, we need to formally specify their expected behaviors. Choosing the specification language is a crucial factor for the overall validation process. We need a language that allows to express easily all desired properties. Moreover the specification language must be easily amenable to validation techniques such as model checking for instance.

Temporal logics [13, 1, 12] are among the most widely used specification languages, the most popular ones being branching time temporal logics (CTL, CTL*) and linear time temporal logic (LTL). These logics are based on pure future modalities, i.e., modalities that does not depend on what happened before the current time. Adding past modalities to LTL does not increase its expressive power [6, 2, 10] but PLTL (LTL with past modalities) is exponentially more succinct than LTL [9]. In this paper, we focus our attention on PLTL.

The drawback of using past modalities is that the satisfiability and the model checking problems are harder to solve. Not harder from a theoretical point of view since both problems are PSPACE complete [15] regardless of whether we use past modalities or not, but harder from a practical point of view. Model checking algorithms for PLTL have already been proposed [7, 14, 18] but few have been implemented and used in actual model checker. This is surprising since past modalities make specifications more succinct and, more importantly, much easier and more natural [11]. We believe that the reason is the difficulty to build efficiently a Büchi automaton associated with a PLTL formula, which is a crucial step for a model checker using PLTL as a specification language. This is precisely the problem addressed in the present paper.

The easiest construction is the so-called *tableau* construction (see e.g. [7, 11, 15]) but a straightforward implementation, called declarative tableau, is highly inefficient. Implementations are based on incremental tableau. They construct only reachable states and apply some simplification techniques. An incremental tableau construction for PLTL [7] requires backtracking and is much harder to implement than for LTL [5].

Following the automata theoretic approach of [17], another technique to generate efficiently a generalized Büchi automaton (GBA) from an LTL formula is to use very-weak alternating automata (VWAA) as an intermediary step. As demonstrated in [3], this yields an implementation which is dramatically faster than those based on the tableau construction. In this paper, we develop the same technique for PLTL. Since we have to deal with past modalities we use two-way very-weak alternating automata (2VWAA). Two-way alternating automata (2AA) were already proposed for specification languages that are much more expressive than PLTL [16, 8] and a translation procedure from general 2AA to GBA is given in [8]. Since PLTL formulas are sufficient to specify most interesting properties and can be easily translated to 2VWAA, it is very important to develop an efficient translation procedure restricted to 2VWAA.

The main result of this paper is an efficient translation procedure of *progressing* 2VWAA to GBA. Starting from a *progressing* 2VWAA with n states, we construct a GBA with at most 2^n states. Since the translation from a PLTL formula φ gives a progressing 2VWAA with at most $|\varphi|+1$ states, we get for this formula a GBA with at most $2^{|\varphi|+1}$ states. As a comparison, the algorithm of [8] gives an automaton with $2^{O(|\varphi|^2)}$ states (recall that the specification language considered in [8] is much more expressive than PLTL).

Due to space constraints, some proofs have had to be omitted in this extended abstract. A full version of the paper is available in [4].

2 Two-way very-weak alternating automata (2VWAA)

A two-way alternating automaton (2AA) is a six-tuple $\mathcal{A} = (Q, \Sigma, \delta, I, F, R)$ where Q is the set of states, Σ is the alphabet, $\delta : Q \to 2^{2^Q \times 2^\Sigma \times 2^Q}$ is the transition function, $I \subseteq 2^Q$ gives the initial condition, $F \subseteq Q$ is the set of final states. $R \subseteq Q$ is the set of repeated states.

A more classical way of defining 2AAs would use $I' \in \mathcal{B}^+(Q)$ and $\delta' : Q \times \Sigma \to \mathcal{B}^+(Q \times \{-1, 1\})$, where -1 and 1 indicate whether the head moves left or right. While the two approaches are equivalent, our definition allows for more compact representations of the automata and for faster algorithms. Notice in particular that in the transition function we use 2^Σ instead of Σ, so that transitions that differ only by actions can be gathered. However, the automaton still reads finite and infinite words in $\Sigma^\infty = \Sigma^+ \cup \Sigma^\omega$.

Runs of 2AAs are defined using Q-forests. A Q-forest over a word $u \in \Sigma^\infty$ is a labeled forest (V, E, σ, ν) where V is the set of vertices, $E \subseteq V \times V$ is the set of edges, $\sigma : V \to Q$ is the state labeling function, $\nu : V \to \mathbb{N}$ is the position labeling function: it indicates the letter the automaton $\mathcal{A}$ is reading; $\forall x \in V$, $0 \leq \nu(x) \leq |u| + 1$ (where $|u| \in \mathbb{N} \cup \{\omega\}$ denotes the length of u), and $\forall (x, y) \in E$, either $\nu(y) = \nu(x) + 1$ if $\mathcal{A}$ goes forward on u, or $\nu(y) = \nu(x) - 1$ if $\mathcal{A}$ goes backward on u.

We shall use the following notations: $\lambda = (\sigma, \nu) : V \to Q \times \mathbb{N}$ is the labelling function, $E(x) = \{y \in V \mid (x, y) \in E\}$ is the set of the sons of x, $\overleftarrow{E}(x) = \{y \in E(x) \mid \nu(y) = \nu(x) - 1\}$ is the set of the left sons of x, and $\overrightarrow{E}(x) = E(x) \backslash \overleftarrow{E}(x)$ is the set of the right sons of x, $\mathrm{left}(x) = E^*(\overleftarrow{E}(x)) \cup \{x\}$, $]x, y] = \{z \in E^+(x) \mid y \in E^*(z)\}$, and similarly for $[x, y]$, $]x, y[$ and $[x, y[$.

A run ρ of a 2AA $\mathcal{A}$ on a word $u = u_1u_2 \cdots \in \Sigma^\infty$ is a Q-forest (V, E, σ, ν) over u such that the roots satisfy the initial condition: if $\Gamma \subseteq V$ are the roots of the forest, then $\sigma(\Gamma) \in I$ and $\nu(\Gamma) \subseteq \{1\}$; and the sons of any node satisfy the transition function: $\forall x \in V$ if $\nu(x) = 0$ or $\nu(x) = |u| + 1$ then $E(x) = \emptyset$, otherwise $\exists \alpha \in 2^\Sigma$, $u_{\nu(x)} \in \alpha$ and $(\sigma(\overleftarrow{E}(x)), \alpha, \sigma(\overrightarrow{E}(x))) \in \delta(\sigma(x))$.

A run ρ is *accepting* if $\forall x \in V$, $\nu(x) = 0$ or $\nu(x) = |u| + 1$ imply $\sigma(x) \in F$; and if any infinite branch of ρ goes infinitely often through R (note that ρ may have infinite branches even if u is finite). Finally, $\mathcal{L}(\mathcal{A})$ is the set of words on which there exists an accepting run of $\mathcal{A}$.

A *two-way very-weak alternating automaton (2VWAA)* is a 2AA where there exists a partial order $\preccurlyeq$ on Q such that $\forall p \in Q$, $\forall (\overleftarrow{X}, \alpha, \overrightarrow{X}) \in \delta(p)$, $\forall q \in \overleftarrow{X} \cup \overrightarrow{X}$, $q \preccurlyeq p$.

Actually, we use *progressing (or loop-free)* 2VWAA for which we describe in Section 4 an efficient translation to Büchi automaton. A run ρ is *progressing* if any infinite branch $x_0 \ldots x_n \ldots$ in ρ satisfies the following property: $\forall N \geq 0$, $\exists i \geq 0$, $\nu(x_i) \geq N$. A run ρ has a *loop* if two nodes on a same branch have the same label: $\exists x \in V$, $\exists y \in E^+(x)$, $\lambda(x) = \lambda(y)$. Otherwise ρ is *loop-free*. A 2AA is *progressing* (respectively *loop-free*) if all runs (accepting or not) of this automaton are progressing (respectively *loop-free*). Note that a loop-free run on a finite word has no infinite branches and is therefore progressing. More generally, any loop-free run is progressing. The converse is trivially false. Still, we have

Proposition 1. *A 2AA is progressing iff it is loop-free.*

Also, given a 2VWAA with n states we can effectively construct a progressing 2VWAA with at most $2n$ states, accepting the same language. When combined with the translation of progressing 2VWAA to GBA presented in Section 4, this yields an efficient translation from general 2VWAA to GBA.

3 PLTL to progressing 2VWAA

The syntax and the semantics of PLTL is classical [12]. Here we use the following syntax: $\varphi ::= \bot \mid p \mid \neg\, \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\, \varphi \mid \mathsf{Y}\, \varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \varphi \,\mathsf{S}\, \varphi$, where $\bot$ stands for *false* and p ranges over the set AP of atomic propositions. An *LTL formula* is a PLTL formula that uses neither Y nor S. Let $\Sigma = 2^{\mathrm{AP}}$. We write $\mathcal{L}(\varphi) \subseteq \Sigma^\infty$ for the set of words satisfying a PLTL formula φ. We also use the dual operators $\top, \wedge, \widetilde{\mathsf{X}}, \widetilde{\mathsf{Y}}, \widetilde{\mathsf{U}}, \widetilde{\mathsf{S}}$. For instance, $\widetilde{\mathsf{X}}\, \varphi = \neg\, \mathsf{X}\, \neg\, \varphi$ and $\varphi_1 \,\widetilde{\mathsf{S}}\, \varphi_2 = \neg(\neg\, \varphi_1 \,\mathsf{S}\, \neg\, \varphi_2)$. Note that $\widetilde{\mathsf{X}}\, \varphi$ is not equivalent to $\mathsf{X}\, \varphi$ on finite words: we have $\widetilde{\mathsf{X}}\, \varphi = \neg\, \mathsf{X} \top \vee \mathsf{X}\, \varphi$.

A PLTL formula can be written in *negative normal form*, where negations are on predicates only. A formula in negative normal form uses $\bot$, $\top$, predicates in

AP, their negations, the operators $\vee$, X, U, Y, S and their duals. A PLTL formula in negative normal form that is neither a disjunction ($\vee$) nor a conjunction ($\wedge$) is called a *temporal formula*. Notice that transforming a formula in negative normal form does not change the number of temporal operators of the formula. From now on, we suppose that every PLTL formula is in negative normal form.

This section is devoted to the first step of our algorithm: from a PLTL formula with n temporal operators we can compute a progressing 2VWAA with at most $n+1$ states. We use the following notations. For $J_1, J_2 \in 2^{2^Q \times 2^\Sigma \times 2^Q}$ we let $J_1 \otimes J_2 = \{(\overleftarrow{X}_1 \cup \overleftarrow{X}_2, \alpha_1 \cap \alpha_2, \overrightarrow{X}_1 \cup \overrightarrow{X}_2) \mid (\overleftarrow{X}_1, \alpha_1, \overrightarrow{X}_1) \in J_1, (\overleftarrow{X}_2, \alpha_2, \overrightarrow{X}_2) \in J_2\}$. For a PLTL formula ψ in negative normal form we define $\overline{\psi}$ inductively by: $\overline{\psi} = \{\{\psi\}\}$ if ψ is a temporal formula, $\overline{\psi_1 \vee \psi_2} = \overline{\psi_1} \cup \overline{\psi_2}$ and $\overline{\psi_1 \wedge \psi_2} = \{X_1 \cup X_2 \mid X_1 \in \overline{\psi_1} \text{ and } X_2 \in \overline{\psi_2}\}$.

It is actually not completely immediate to deal smoothly with the special cases raised when checking a past modality at the beginning of a word, or a future modality at the end of a finite word. We solve this problem using a special state END, which is reached in an accepting run when the current position ν is outside the word u ($\sigma(x) = \text{END} \Rightarrow \nu(x) = 0$ or $\nu(x) = |u| + 1$).

Definition 2 (φ to $\mathcal{A}_\varphi$). *For any PLTL formula φ in negative normal form on the set* AP*, let $\mathcal{A}_\varphi = (Q, \Sigma, \delta, I, F, R)$ be the 2AA defined by $Q = sub(\varphi) \cup \{\text{END}\}$ where $sub(\varphi)$ are the temporal subformulae of φ, $\Sigma = 2^{\text{AP}}$, $I = \overline{\varphi}$, $F = \{\text{END}\}$, R is the set of the subformulae in $sub(\varphi)$ that are not of the form $\varphi_1 \mathsf{U} \varphi_2$, and δ is defined below (Δ extends δ to $\mathcal{B}^+(Q)$):*

$$\begin{cases}
\delta(\bot) = \emptyset \\
\delta(\top) = \{(\emptyset, \Sigma, \emptyset)\} \\
\delta(p) = \{(\emptyset, \Sigma_p, \emptyset)\} \text{ where } \Sigma_p = \{a \in \Sigma \mid p \in a\} \\
\delta(\neg p) = \{(\emptyset, \Sigma_{\neg p}, \emptyset)\} \text{ where } \Sigma_{\neg p} = \Sigma \backslash \Sigma_p \\
\delta(\mathsf{X}\,\psi) = \{(\emptyset, \Sigma, e) \mid e \in \overline{\psi}\} \\
\delta(\widetilde{\mathsf{X}}\,\psi) = \{(\emptyset, \Sigma, e) \mid e \in \overline{\psi}\} \cup \{(\emptyset, \Sigma, \{\text{END}\})\} \\
\delta(\mathsf{Y}\,\psi) = \{(e, \Sigma, \emptyset) \mid e \in \overline{\psi}\} \\
\delta(\widetilde{\mathsf{Y}}\,\psi) = \{(e, \Sigma, \emptyset) \mid e \in \overline{\psi}\} \cup \{(\{\text{END}\}, \Sigma, \emptyset)\} \\
\delta(\psi_1 \,\mathsf{U}\, \psi_2) = \Delta(\psi_2) \ \cup (\Delta(\psi_1) \otimes \{(\emptyset, \Sigma, \{\psi_1 \,\mathsf{U}\, \psi_2\})\}) \\
\delta(\psi_1 \,\widetilde{\mathsf{U}}\, \psi_2) = \Delta(\psi_2) \ \otimes (\Delta(\psi_1) \cup \{(\emptyset, \Sigma, \{\psi_1 \,\widetilde{\mathsf{U}}\, \psi_2\}), (\emptyset, \Sigma, \{\text{END}\})\}) \\
\delta(\psi_1 \,\mathsf{S}\, \psi_2) = \Delta(\psi_2) \ \cup (\Delta(\psi_1) \otimes \{(\{\psi_1 \,\mathsf{S}\, \psi_2\}, \Sigma, \emptyset)\}) \\
\delta(\psi_1 \,\widetilde{\mathsf{S}}\, \psi_2) = \Delta(\psi_2) \ \otimes (\Delta(\psi_1) \cup \{(\{\psi_1 \,\widetilde{\mathsf{S}}\, \psi_2\}, \Sigma, \emptyset), (\{\text{END}\}, \Sigma, \emptyset)\}) \\
\delta(\text{END}) = \emptyset
\end{cases}$$

$$\begin{cases}
\Delta(\psi) = \delta(\psi) \text{ if } \psi \text{ is a temporal formula} \\
\Delta(\psi_1 \vee \psi_2) = \Delta(\psi_1) \cup \Delta(\psi_2) \\
\Delta(\psi_1 \wedge \psi_2) = \Delta(\psi_1) \otimes \Delta(\psi_2)
\end{cases}$$

Theorem 3 (PLTL to progressing 2VWAA). *For any PLTL formula in negative normal form φ with n temporal subformulae, the automaton $\mathcal{A}_\varphi$ is a progressing 2VWAA with at most $n+1$ states and $\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}(\varphi)$.*

Proof. Let us define $\psi_1 \preccurlyeq \psi_2$ if $\psi_1 = \text{END}$ or if ψ_1 is a subformula of ψ_2. It is easy to see with this partial order that $\mathcal{A}_\varphi$ is very-weak.

Now, we show that $\mathcal{A}_\varphi$ is progressing. Suppose that a node x and its son y have the same state-label ψ in a run of $\mathcal{A}_\varphi$. Then, either ($y \in \overrightarrow{E}(x)$ and ($\psi = \psi_1 \mathsf{U} \psi_2$ or $\psi = \psi_1 \widetilde{\mathsf{U}} \psi_2$)), or ($y \in \overleftarrow{E}(x)$ and ($\psi = \psi_1 \mathsf{S} \psi_2$ or $\psi = \psi_1 \widetilde{\mathsf{S}} \psi_2$)).

Now let $x_0, x_1, \ldots$ be an infinite branch of a run ρ of $\mathcal{A}_\varphi$. Since $\mathcal{A}_\varphi$ is very-weak, the sequence $\sigma(x_0), \sigma(x_1), \ldots$ is ultimately constant. From the previous statement, necessarily the limit of this sequence is a formula $\psi_1 \mathsf{U} \psi_2$ or $\psi_1 \widetilde{\mathsf{U}} \psi_2$, and hence ν is strictly increasing from a certain node on the branch: $\mathcal{A}_\varphi$ is progressing.

The proof for $\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}(\varphi)$ is omitted. It is similar to the classical proof used for the similar translation of LTL formulas to VWAA. $\square$

4 Progressing 2VWAA to GBA

The general translation from a 2AA to a BA is rather involved. Here we take full advantage of the fact that we start from a progressing 2VWAA. The basic idea, starting from an accepting run ρ of a 2AA $\mathcal{A}$ over a word $u = u_1 u_2 \ldots \in \Sigma^\infty$, is to consider the sets $X_i = \sigma(\nu^{-1}(i))$, and to build a kind of automaton $\mathcal{B}$ such that the sequence $X_0, X_1, X_2, \ldots$ is an accepting run of $\mathcal{B}$ on u. Transitions of $\mathcal{B}$ should reflect the fact that each node x of ρ satisfies the transition function of $\mathcal{A}$. If $\nu(x) = i$, this involves the sets X_{i-1}, X_i, X_{i+1} and the letter u_i. Hence it is natural to consider quadruples $(X_{i-1}, X_i, X_{i+1}, u_i)$ as transitions of $\mathcal{B}$. This is why we introduce here *generalized* BA having such quadruples as transitions. We are also using several acceptance conditions on transitions instead of states since the construction becomes simpler.

A GBA is a five-tuple $\mathcal{G} = (Q, \Sigma, T, I, F, \mathcal{T})$ where: Q is the set of states, Σ is the alphabet, $T \subseteq Q \times Q \times Q \times 2^\Sigma$ is the transition function, $I \subseteq Q \times Q$ is the set of initial states, $F \subseteq Q \times Q$ is the set of final states, $\mathcal{T} = \{T_1, \ldots, T_r\}$ where $T_j \subseteq T$ are the acceptance tables.

A *run* σ of a GBA $\mathcal{G}$ on a word $u = u_1 u_2 \ldots \in \Sigma^\omega$ is an infinite sequence $q_0, q_1, \alpha_1, \ldots, q_n, \alpha_n, \ldots$ such that $(q_0, q_1) \in I$, and $\forall i \geq 1$, $u_i \in \alpha_i$ and $(q_{i-1}, q_i, q_{i+1}, \alpha_i) \in T$. A run on a finite word $u \in \Sigma^+$ is defined similarly. The run σ is *accepting* if $u \in \Sigma^n$ and $(q_n, q_{n+1}) \in F$ or $u \in \Sigma^\omega$ and σ uses infinitely many transitions from each T_j, $1 \leq j \leq r$. $\mathcal{L}(\mathcal{G})$ is the set of words on which there exists an accepting run of $\mathcal{G}$.

Definition 4 ($\mathcal{A}$ to $\mathcal{G}^1_\mathcal{A}$). *For any progressing 2VWAA $\mathcal{A} = (Q, \Sigma, \delta, I, F, R)$, we define the GBA $\mathcal{G}^1_\mathcal{A} = (Q', \Sigma, T', I', F', \mathcal{T}')$ by $Q' = 2^Q$,*
$T' = \{(\overleftarrow{X}, X, \overrightarrow{X}, \alpha) \mid \exists\ (\overleftarrow{Y}, \alpha, \overrightarrow{Y}) \in \bigotimes_{q \in X} \delta(q)$ *with* $\overleftarrow{Y} \subseteq \overleftarrow{X}$ *and* $\overrightarrow{Y} \subseteq \overrightarrow{X}\}$,
$I' = \{(X, Y) \in Q' \times Q' \mid X \subseteq F$ *and* $\exists Z \in I$ *with* $Z \subseteq Y\}$,
$F' = \{(X, Y) \in Q' \times Q' \mid Y \subseteq F\}$, *and* $\mathcal{T}' = \{T'_q \mid q \in Q \backslash R\}$ *where*
$T'_q = \{(\overleftarrow{X}, X, \overrightarrow{X}, \alpha) \in T' \mid q \notin X$ *or*
$\exists (\overleftarrow{Y}, \beta, \overrightarrow{Y}) \in \delta(q)$ *with* $\overleftarrow{Y} \subseteq \overleftarrow{X}$, $\overrightarrow{Y} \subseteq \overrightarrow{X}$, $\beta \supseteq \alpha$ *and* $q \notin \overrightarrow{Y}\}$

Indeed, the hard part in a translation from a 2AA $\mathcal{A}$ to a GBA $\mathcal{G}$ is to make sure that each accepting run of $\mathcal{G}$, which is a *flat* sequence of nodes, can be lifted to a Q-forest which defines an accepting run of $\mathcal{A}$. It is remarkable that if $\mathcal{A}$ is a progressing 2VWAA, we can simply use sequences $X_0, X_1, X_2, \ldots$ as runs of $\mathcal{G}$ and still be able to lift them to accepting runs of $\mathcal{A}$.

Proposition 5. *For any progressing 2VWAA $\mathcal{A}$, $\mathcal{L}(\mathcal{G}^1_{\mathcal{A}}) \subseteq \mathcal{L}(\mathcal{A})$.*

Actually $\mathcal{L}(\mathcal{G}^1_{\mathcal{A}}) = \mathcal{L}(\mathcal{A})$, but the other inclusion follows directly from results that will be proved afterwards, hence we will not prove it here. See theorem 9 for the final result.

Proof. Let $X_0, X_1, \alpha_1, X_2, \alpha_2, \ldots$ be an accepting run of $\mathcal{G}^1_{\mathcal{A}}$ on a word $u = u_1u_2\ldots \in \Sigma^\infty$. We are going to define a labelled graph $\rho = (V, E, \sigma, \nu)$. Let $V = \{(q, i) \mid i \geq 0,\ q \in X_i\}$. By definition of T', $\forall 1 \leq i \leq |u|$, $\forall q \in X_i$, there exist $\overleftarrow{Y} \subseteq X_{i-1}$, $\overrightarrow{Y} \subseteq X_{i+1}$ and $\alpha \in 2^\Sigma$ with $\alpha_i \subseteq \alpha$ and $(\overleftarrow{Y}, \alpha, \overrightarrow{Y}) \in \delta(q)$, and such that $q \notin \overrightarrow{Y}$ whenever $q \notin R$ and $(X_{i-1}, X_i, X_{i+1}, \alpha_i) \in T'_q$. Let $E((q, i)) = \overleftarrow{Y} \times \{i-1\} \cup \overrightarrow{Y} \times \{i+1\}$. If $i \in \{0, |u|+1\}$ then let $E((q, i)) = \emptyset$. $\forall (q, i) \in V$, let $\sigma((q, i)) = q$ and $\nu((q, i)) = i$.

Since $(X_0, X_1) \in I'$, $\exists \Gamma \subseteq X_1 \times \{1\}$ such that $\sigma(\Gamma) \in I$. Now if we "unfold" the labelled graph $\rho = (V, E, \sigma, \nu)$ from the set of roots Γ, we obtain a Q-forest $\rho' = (V', E', \sigma', \nu')$.

Let $x \in V'$. If $\nu'(x) \in \{0, |u|+1\}$ then $E'(x) = \emptyset$. Otherwise, let $q = \sigma'(x)$ and $i = \nu'(x)$, let $\overleftarrow{Y} = \sigma'(\overleftarrow{E'}(x)) = \sigma(E(q, i) \cap Q \times \{i-1\})$ and $\overrightarrow{Y} = \sigma'(\overrightarrow{E'}(x)) = \sigma(E(q, i) \cap Q \times \{i+1\})$. By construction of ρ there exists $\alpha \in 2^\Sigma$ with $\alpha_i \subseteq \alpha$ and $(\overleftarrow{Y}, \alpha, \overrightarrow{Y}) \in \delta(q)$. Since $u_i \in \alpha_i$, we have proved that the sons of x satisfy the transition function. Hence ρ' is a run of $\mathcal{A}$ on u.

We shall now check that ρ' is accepting:

- $\sigma'(\nu'^{-1}(0)) \subseteq \sigma(\nu^{-1}(0)) = X_0 \subseteq F$, since $(X_0, X_1) \in I'$,
- if u is finite of length n then $\sigma'(\nu'^{-1}(n+1)) \subseteq \sigma(\nu^{-1}(n+1)) = X_{n+1} \subseteq F$ since $(X_n, X_{n+1}) \in F'$,
- assume that ρ' contains an infinite branch $x_0, x_1, \ldots$ ultimately state-labelled in $Q \backslash R$: $\exists q \in Q \backslash R$, $\exists N > 0$, $\forall i \geq N$, $\sigma(x_i) = q$. Since $\mathcal{A}$ is progressing, from proposition 1 we know that ρ' is loop-free (which implies that ρ is loop-free) and u is infinite. Since $X_0, X_1, \ldots$ is accepting, $\exists k \geq \nu(x_N)$ such that $(X_{k-1}, X_k, X_{k+1}, \alpha_k) \in T'_q$. ρ' is loop-free so ν is necessarily strictly increasing on $x_N, x_{N+1}, \ldots$, and there exists $j \geq N$ such that $\lambda(x_j) = (q, k)$ and $\lambda(x_{j+1}) = (q, k+1)$. But since $\sigma(x_j) = q$ and $(X_{k-1}, X_k, X_{k+1}, \alpha_k) \in T'_q$ we chose $E((q, k))$ in ρ such that $(q, k+1) \notin E((q, k))$. This is a contradiction, and hence ρ' cannot contain such a branch. $\square$

It is quite easy to see that $\mathcal{G}^1_{\mathcal{A}}$ is still too big to be used in an efficient implementation. It contains many useless states, and keeping only the accessible states would not be enough since the initial states are too numerous already. We introduced $\mathcal{G}^1_{\mathcal{A}}$ merely to prove more easily that our construction is correct.

The intuition how to get a smaller GBA from a 2VWAA is as follows: by removing useless parts of runs of a 2VWAA we obtain *minimal runs*, such that removing any subtree of the forest makes the run invalid. Ideally, we would like to construct a GBA $\mathcal{G}^2_{\mathcal{A}}$ which is the restriction of $\mathcal{G}^1_{\mathcal{A}}$ to transitions of the form $(\sigma(\nu^{-1}(i-1)), \sigma(\nu^{-1}(i)), \sigma(\nu^{-1}(i+1)), u_i)$ obtained from minimal runs.

For this, we start from a small set of initial states, and we compute the states and transitions accessible from these states. We need to store both the current set of states Y and the previous one X. Since $\mathcal{A}$ is two-way, it may happen that the set X is not big enough to fulfil all requirements imposed by Y. In this case, we have to backtrack and enlarge X.

Definition 6 ($\mathcal{A}$ to $\mathcal{G}^2_{\mathcal{A}}$). *For any progressing 2VWAA $\mathcal{A} = (Q, \Sigma, \delta, I, F, R)$, let $\mathcal{G}^2_{\mathcal{A}} = (Q'', \Sigma, T'', I'', F'', \mathcal{T}'')$ be the GBA computed by*
Initialization: $I'' = \{F\} \times I$, $\nabla = \{F\} \times I$, $T'' = \emptyset$. Then, we apply the following saturation procedure for each state $(X, Y) \in \nabla$ until we reach a fixed point:

`for each` $(X', \alpha, Z) \in \bigotimes_{q \in Y} \delta(q)$

  `if` $X' \subseteq X$
(a)   `if` $(Y, Z) \notin \nabla$ `then add` (Y, Z) `to` ∇
       `if` $(X, Y, Z, \alpha) \notin T''$ `then add` (X, Y, Z, α) `to` T''
  `else`
    `for each` $(F, X, Y, \beta) \in T''$ `with` $(F, X) \in I''$
(b)   `if` $(F, X \cup X') \notin \nabla$ `then add` $(F, X \cup X')$ `to` ∇
      `add` $(F, X \cup X')$ `to` I''
    `for each` $(V, W, X, \gamma), (W, X, Y, \beta) \in T''$
(c)   `if` $(W, X \cup X') \notin \nabla$ `then add` $(W, X \cup X')$ `to` ∇
      `if` $(V, W, X \cup X', \gamma) \notin T''$ `then add` $(V, W, X \cup X', \gamma)$ `to` T''

Finally, we set $Q'' = \{X \in 2^Q \mid \exists Y \in 2^Q,\ (X, Y) \in \nabla \text{ or } (Y, X) \in \nabla\}$
$F'' = (2^Q \times 2^F) \cap (Q'' \times Q'')$, and $\mathcal{T}'' = \{T''_q \mid q \in Q \backslash R\}$ where

$$T''_q = \{(\overleftarrow{X}, X, \overrightarrow{X}, \alpha) \in T'' \mid q \notin X \text{ or } \exists(\overleftarrow{Y}, \beta, \overrightarrow{Y}) \in \delta(q) \text{ with } \overleftarrow{Y} \subseteq \overleftarrow{X},\ \overrightarrow{Y} \subseteq \overrightarrow{X},\ \beta \supseteq \alpha \text{ and } q \notin \overrightarrow{Y}\}$$

Proposition 7. *For any progressing 2VWAA $\mathcal{A}$, $\mathcal{L}(\mathcal{G}^2_{\mathcal{A}}) \subseteq \mathcal{L}(\mathcal{G}^1_{\mathcal{A}})$.*

Proof. It is easy to notice that any state or transition of $\mathcal{G}^2_{\mathcal{A}}$ appears in $\mathcal{G}^1_{\mathcal{A}}$: $Q'' \subseteq Q'$ and $T'' \subseteq T'$. Hence any accepting run of $\mathcal{G}^2_{\mathcal{A}}$ on a word u is also an accepting run of $\mathcal{G}^1_{\mathcal{A}}$ on the same word. □

Proposition 8. *For any progressing 2VWAA $\mathcal{A}$, $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{G}^2_{\mathcal{A}})$.*

Proof. Let $\rho = (V, E, \sigma, \nu)$ be an accepting run of $\mathcal{A}$ on a word u. We build by induction on k a sequence $\rho'_k = X_0, (X_i, \alpha_i)_{0<i<n}, X_n$ with $1 \leq n \leq |u| + 1$ and such that $(X_0, X_1) \in I''$, $X_i \subseteq \sigma(\nu^{-1}(i))$ and $(X_{i-1}, X_i) \in \nabla$ for all $1 \leq i \leq n$, and for all $1 \leq i < n$, $u_i \in \alpha_i$ and $(X_{i-1}, X_i, X_{i+1}, \alpha_i) \in T''$ and $\forall q \in X_i \backslash R$, if $\exists x \in \nu^{-1}(i)$ with $\sigma(x) = q$ and $q \notin \sigma(\overrightarrow{E}(x))$ then $(X_{i-1}, X_i, X_{i+1}, \alpha_i) \in T''_q$.

Let us start with $\rho'_0 = X_0, X_1$ with $X_0 = F$ and $X_1 = \sigma(\Gamma)$ where Γ are the roots of ρ. Since $X_1 \in I$ then $(X_0, X_1) \in I'' \subseteq \nabla$, the inductive hypothesis is satisfied. Now suppose we have built a sequence $\rho'_k = X_0, (X_i, \alpha_i)_{0<i<n}, X_n$ for some $1 \leq n \leq |u|$, satisfying the inductive hypothesis. If $n = |u| + 1$ we stop the construction.

Otherwise, $\forall q \in X_n$, choose $x_q \in V$ such that $\lambda(x_q) = (q, n)$, and such that if $q \notin R$ then $q \notin \sigma(\overrightarrow{E}(x_q))$ whenever this is possible. Let $X = \{x_q \mid q \in X_n\}$, $X'_{n-1} = \sigma(\overleftarrow{E}(X))$, $X_{n+1} = \sigma(\overrightarrow{E}(X))$. Since ρ is a run of $\mathcal{A}$ on u and $n \leq |u|$, $\exists \alpha_n$ with $u_n \in \alpha_n$ such that $(X'_{n-1}, \alpha_n, X_{n+1}) \in \bigotimes_{q \in X_n} \delta(q)$. Since ρ'_k satisfies the inductive hypothesis, we have $(X_{n-1}, X_n) \in \nabla$ and three cases can occur:

(a) if $X'_{n-1} \subseteq X_{n-1}$, then from the construction of $\mathcal{G}^2_{\mathcal{A}}$, $(X_n, X_{n+1}) \in \nabla$ and $(X_{n-1}, X_n, X_{n+1}, \alpha_n) \in T''$. Moreover, for each $q \in X_n \setminus R$, if $\exists x \in \nu^{-1}(n)$ with $\sigma(x) = q$ and $q \notin \sigma(\overrightarrow{E}(x))$ then $\exists \beta \supseteq \alpha_n$, $\exists X' \subseteq X_{n-1}$, $\exists Z' \subseteq X_{n+1} \backslash \{q\}$, $(X', \beta, Z') \in \delta(q)$ hence $(X_{n-1}, X_n, X_{n+1}, \alpha_n) \in T''_q$. Therefore, $\rho'_{k+1} = \rho'_k, \alpha_n, X_{n+1}$ satisfies the inductive hypothesis.

We now suppose that $X'_{n-1} \nsubseteq X_{n-1}$. Note that this implies $n \geq 2$, because if $n = 1$ then since ρ is accepting, $X'_0 \subseteq F = X_0$.

(b) If $n = 2$ then $(X_0, X_1, X_2, \alpha_1) \in T''$ and $(X_0, X_1) \in I''$ so from the construction of $\mathcal{G}^2_{\mathcal{A}}$, $(X_0, X_1 \cup X'_1) \in I'' \subseteq \nabla$. Hence $\rho'_{k+1} = X_0, X_1 \cup X'_1$ satisfies the inductive hypothesis.

(c) Assume now that $n \geq 3$. Then we have $(X_{n-3}, X_{n-2}, X_{n-1}, \alpha_{n-2}) \in T''$ and $(X_{n-2}, X_{n-1}, X_n, \alpha_{n-1}) \in T''$. From the construction of $\mathcal{G}^2_{\mathcal{A}}$, $(X_{n-3}, X_{n-2}, X_{n-1} \cup X'_{n-1}, \alpha_{n-2}) \in T''$ and $(X_{n-2}, X_{n-1} \cup X'_{n-1}) \in \nabla$. Moreover if $(X_{n-3}, X_{n-2}, X_{n-1}, \alpha_{n-2}) \in T''_q$ then $(X_{n-3}, X_{n-2}, X_{n-1} \cup X'_{n-1}, \alpha_{n-2}) \in T''_q$. Therefore, $\rho'_{k+1} = X_0, \ldots, X_{n-2}, \alpha_{n-2}, X_{n-1} \cup X'_{n-1}$ satisfies the inductive hypothesis.

Now, we show that the sequence $(\rho'_k)_{k \geq 0}$ converges. For this, consider the alphabet $A = 2^Q \times 2^\Sigma$ partially ordered by $(X, \alpha) \preccurlyeq (Y, \beta)$ if $X \subseteq Y$. Let us write $\rho'_k = (X_0, \emptyset), (X_i, \alpha_i)_{1 \leq i \leq n}, (X_{n+1}, \emptyset) \in A^*$. The sequence of words $(\rho'_k)_{k \geq 0}$ is strictly increasing for the lexicographic order induced by $\preccurlyeq$. Hence, we can easily show that this sequence is either finite (if the construction terminates) or infinite and converges to an infinite word $\rho' \in A^\omega$.

If u is finite then the words ρ'_k are of length at most $|u| + 2$. Therefore the construction terminates at some step k: let $\rho' = \rho'_k$. Since ρ is accepting, $\sigma(\nu^{-1}(|u| + 1)) \subseteq F$ so $(X_{|u|}, X_{|u|+1}) \in F''$. Hence ρ' is an accepting run of $\mathcal{G}^2_{\mathcal{A}}$ on u.

If u is infinite then the sequence $(\rho'_k)_{k \geq 0}$ must be infinite and converges to an infinite word $\rho' \in A^\omega$. Since any prefix of ρ' is also a prefix of ρ_k for some k, ρ' is a run of $\mathcal{G}^2_{\mathcal{A}}$ on u. Suppose that ρ' is not accepting: there exists $q \in Q \backslash R$ such that after a given rank N, all transitions used in ρ' are not in T''_q. Hence $\forall x \in V$ if $\lambda(x) = (q, i)$ with $i \geq N$ then $q \in \sigma(\overrightarrow{E}(x))$. Moreover, since $(X_{N-1}, X_N, X_{N+1}, \alpha_N) \notin T''_q$, we have $q \in X_N$. So we can find in ρ an infinite branch ultimately labelled by q, which is a contradiction with ρ accepting. $\square$

As a consequence of Propositions 5, 7 and 8 we obtain:

Theorem 9 (progressing 2VWAA to GBA). *For any progressing 2VWAA $\mathcal{A}$, $\mathcal{L}(\mathcal{G}^1_{\mathcal{A}}) = \mathcal{L}(\mathcal{G}^2_{\mathcal{A}}) = \mathcal{L}(\mathcal{A})$. Hence from any progressing 2VWAA with n states and r repeated states we can construct a GBA accepting the same language with at most 2^n states and $n - r$ acceptance tables.*

We conclude this section by explaining several simplifications that are used in the implementation of the algorithm described in Definition 6.
Saturation procedure: In order to generate all needed transitions, we have to repeat the saturation procedure until no more changes occur. This has to be carefully implemented to avoid redundant computations. The idea here is to timestamp states and transitions by integers. When they are first created, new elements of ∇ are timestamped by 0, whereas new transitions in T'' get the current timestamp. Each time a pair (X, Y) is considered for saturation, the current timestamp is incremented, and when this pair has been treated, it gets the current timestamp. Now, step `(a)` is executed only when the timestamp of (X, Y) is 0. In step `(b)` we consider only transitions (F, X, Y, β) whose timestamps are greater or equal to that of (X, Y). Similarly, in step `(c)`, we only consider pairs of transitions for which at least one timestamp is greater or equal to that of (X, Y).
Transition simplification: The idea here is to remove redundant transitions. When there exist two transitions (X, Y, Z, α) and (X, Y, Z, β) with $\alpha \subseteq \beta$, the first transition is more restrictive than the second one, and thus can be deleted. This simplification can be done *on-the-fly*, that is when the algorithm adds a new transition to T''.
State simplification: The idea here is to merge equivalent pairs of states. When two pairs of states (X, Y) and (X', Y') have the same outgoing transitions $((X, Y, Z, \alpha) \in T'' \iff (X', Y', Z, \alpha) \in T''$ and $\forall q \in Q\backslash R$, $(X, Y, Z, \alpha) \in T''_q \iff (X', Y', Z, \alpha) \in T''_q)$, they can be merged, and any transition pointing on one of the two states can now point on the merged state.

The overall simplification alternates transition simplification and state simplification until a fixed point is reached.

Experimental results: We have implemented two algorithms, building respectively $\mathcal{G}^1_{\mathcal{A}}$ and $\mathcal{G}^2_{\mathcal{A}}$ from a PLTL formula φ. Here are the results of some computations on the formula $\varphi_n = \neg\, \mathsf{G}(p_1 \rightarrow \mathsf{F}^{-1}(p_2 \wedge \mathsf{F}^{-1}(p_3 \wedge \ldots \mathsf{F}^{-1}\, p_n) \ldots))$, stating that each p_1 is preceded by p_2, preceded itself by p_3, and so on until p_n. This demonstrates the striking improvement of the on-the-fly algorithm.

	$\mathcal{A}_\varphi$	$\mathcal{G}^1_{\mathcal{A}}$				$\mathcal{G}^2_{\mathcal{A}}$			
	states	before	after	time	space	before	after	time	space
φ_2	3	28, 82	6, 11	0.03	<380	7, 10	4, 6	0.01	<380
φ_3	4	100, 544	12, 36	0.83	<380	10, 19	6,13	0.01	<380
φ_4	5	364, 3630	27,102	230	1,700	13, 31	8,23	0.08	635
φ_5	6	1348,24830	58,264	130,000	39,000	16, 46	10,36	9.40	32,000

In order of appearance : tested formula, number of states of the VWAA, and for each $i \in \{1, 2\}$, number of states and transitions of $\mathcal{G}^i_{\mathcal{A}}$ *before* and *after* simplification, computation time in seconds, memory used in KB.

Model-checking: At this point we are able from any PLTL formula φ to compute a GBA accepting the models of φ. In order to apply usual model checking techniques we can transform our GBA $\mathcal{G} = (Q, \Sigma, T, I, F, \mathcal{T})$ to a more conventional automaton $\mathcal{G}' = (Q', \Sigma, T', I, F, \mathcal{T}')$ where $T' = \{((q_-, q), \alpha, (q, q_+)) \mid$

$(q_-, q, q_+, \alpha) \in T\}$, the same for acceptance tables, and Q' is the set of all pairs (q_-, q) or (q, q_+) appearing in T'. We can actually view $\mathcal{G}$ as a convenient encoding of $\mathcal{G}'$. Classical techniques allows also to get a Büchi automaton (BA) replacing the acceptance tables by a single set of repeated states.

References

1. E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 16, pages 995–1072. Elsevier Science, 1990.
2. D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. of PoPL'80*, pages 163–173, Las Vegas, Nev., 1980.
3. P. Gastin and D. Oddoux. Fast ltl to büchi automata translation. In *Proc. of CAV'01*, number 2102 in LNCS, pages 53–65. Springer Verlag, 2001.
4. P. Gastin and D. Oddoux. LTL with past and two-way very-weak alternating automata. Tech. Rep. LIAFA 2003–010, Université Paris 7 (France).
5. R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification Testing and Verification*, pages 3–18, Warsaw, Poland, 1995. Chapman & Hall.
6. J.A.W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, California, 1968.
7. Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. A decision algorithm for full propositional temporal logic. In *Proc. of CAV'93*, number 697 in LNCS, pages 97–109. Springer Verlag, 1993.
8. O. Kupferman, N. Piterman, and M.Y. Vardi. Extended temporal logic revisited. In *Proc. of CONCUR'01*, number 2154 in LNCS, pages 519–535. Springer Verlag, 2001.
9. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *Proc. of LICS'02*, pages 383–392, 2002.
10. F. Laroussinie and Ph. Schnoebelen. A hierarchy of temporal logics with past. *Theoretical Computer Science*, 148:303–324, 1995.
11. O. Lichtenstein, A. Pnueli, and L.D. Zuck. The glory of the past. In *Proc. of the 3rd Workshop on Logics of Programs*, number 193 in LNCS, pages 196–218. Springer Verlag, 1985.
12. Z. Manna and A. Pnueli. *The temporal logic of reactive and concurent systems: Specification*. Springer Verlag, Berlin-Heidelberg-New York, 1992.
13. A. Pnueli. The temporal logic of programs. In *Proc. of FOCS'77*, pages 46–57, 1977.
14. Y.S. Ramakrishna, L.E. Moser, L.K. Dillon, P.M. Melliar-Smith, and G. Kutty. An automata theoretic decision procedure for propositional temporal logic with Since and Until. *Fundamenta Informaticae*, 17:271–282, 1992.
15. A.P. Sistla and E.M. Clarke. The complexity of propositional linear time logic. *Journal of the Association of Computing Machinery*, 32:733–749, 1985.
16. M.Y. Vardi. Reasonning about the past with two-way automata. In *Proc. of ICALP'98*, number 1443 in LNCS, pages 628–641. Springer Verlag, 1998.
17. M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. of LICS'86*, pages 332–344, 1986.
18. M.Y. Vardi and P. Wolper. Reasonning about infinite computations. *Information and Computation*, 115:1–37, 1994.