Deeper Connections between LTL and Alternating Automata

Radek Pelánek*[1] and Jan Strejček**[1]

Faculty of Informatics, Masaryk University in Brno
Botanická 68a, 602 00 Brno, Czech Republic
{xpelanek,strejcek}@fi.muni.cz

Abstract. It is known that Linear Temporal Logic (LTL) has the same expressive power as alternating 1-weak automata (A1W automata, also called alternating linear automata or very weak alternating automata). A translation of LTL formulae into a language equivalent A1W automata has been introduced in [1]. The inverse translation has been developed independently in [2] and [3]. In the first part of the paper we show that the latter translation wastes temporal operators and we propose some improvements of this translation. The second part of the paper draws a direct connection between fragments of the Until-Release hierarchy [4] and alternation depth of nonaccepting and accepting states in A1W automata. We also indicate some corollaries and applications of these results.

1 Introduction

The study of connections between temporal logics and automata proved to be very fruitful. The best example is the translation of *linear temporal logic (LTL)* formulae into nondeterministic Büchi automata [5, 6], which is one of the cornerstones of the automata-based model checking of LTL properties [7].

It is known for a long time that nondeterministic Büchi automata are more expressive than LTL [8]. Only a few years ago, the *alternating 1-weak Büchi automata* (or *A1W automata* for short, also known as *alternating linear automata* or *very weak alternating automata*) have been identified as the type of automata with the same expressive power as LTL. Muller, Saoudi, and Schupp [1] have introduced a translation of LTL formulae into equivalent A1W automata. The translation of A1W automata into equivalent LTL formulae has been presented independently by Rohde [2], and Löding and Thomas [3].

The LTL→A1W translation has been recently used to build new and more efficient algorithms translating LTL formulae into nondeterministic Büchi automata [9, 10]. Another application of this translation arises in connection with verification algorithms working directly on alternating automata (for pointers see [11]). The growing popularity of A1W automata is hindered by the fact that

* Supported by GAČR, grants No. 201/03/0509 and 102/05/H050.
** Supported by Grant Agency of the ASCR, grant No. 1ET408050503.

it is often hard to see what language is recognized by an automaton. Here is the point where the A1W→LTL translation can help as LTL formulae are easy to understand, especially if they contain only few occurrences of temporal operators. Unfortunately, the "standard" A1W→LTL translation does not provide optimal results as it wastes *next* operators. For example, the automaton corresponding to the formula $a \mathsf{U} (b \wedge (b \mathsf{U} c))$ is translated into formula $a \mathsf{U} (b \wedge \mathsf{X}(b \mathsf{U} c))$. In this paper we propose an improved A1W→LTL translation reducing the number of *next* operators in the resulting formula. Our improved translation also prefers the use of less expressive and easy-to-read unary temporal operators *eventually* or *globally* instead of binary operator *until*. We prove that for an A1W automaton produced by the standard translation of an LTL formula φ our translation provides a formula with the same (or even lower) nesting depths of *until*, *next*, and *eventually* operators comparing to these nesting depths in φ.

The improved translation also allows to define classes of A1W automata with the same expressive power as LTL fragments with temporal operators *until*, *next*, and *eventually*, where the nesting depth of each operator can be bounded. Several interesting and previously studied LTL fragments fit into this general pattern, namely fragments of the *until hierarchy* [12, 13], fragments without *eventually* operator and with bounded nesting depth(s) of *next* or *until* or both operators studied in [14, 15], and the fragment without *until* operator known as *restricted LTL* [16].

The second part of this paper presents connections between A1W automata and some LTL fragments that are not covered by the pattern above, namely fragments of the *until-release (alternating) hierarchy* [4] and fragments of the *hierarchy of temporal properties* [17, 18]. In particular, we show that alternation of *until* and *release* operators in a formula corresponds to alternation of nonaccepting and accepting states in an equivalent A1W automaton. Some corollaries of this correspondence are presented as well.

The paper is structured as follows. In Section 2 we recall the definitions of LTL and alternating 1-weak automata together with standard translations between these formalisms. Section 3 provides an improved version of A1W→LTL translation and indicates some applications. Section 4 is devoted to the connection between A1W automata and the until-release hierarchy. Section 5 sums up presented results and mentions some topics for future research. All proofs are omitted due to the space limitations; they can be found in the full version of this paper [19].

2 Preliminaries

2.1 Linear Temporal Logic (LTL)

The syntax of LTL is given by the abstract syntax equation

$$\varphi \ ::= \ \top \ \mid \ a \ \mid \ \neg\varphi \ \mid \ \varphi_1 \wedge \varphi_2 \ \mid \ \mathsf{X}\varphi \ \mid \ \mathsf{F}\varphi \ \mid \ \varphi_1 \mathsf{U} \varphi_2,$$

where $\top$ stands for *true* and a ranges over a countable set $\Lambda = \{a, b, c, \ldots\}$ of *letters*. We also use $\perp$ to abbreviate $\neg\top$, $\mathsf{G}\varphi$ to abbreviate $\neg\mathsf{F}\neg\varphi$, and $\varphi \mathsf{R} \psi$

to abbreviate $\neg(\neg\varphi \mathsf{U} \neg\psi)$. The temporal operators $\mathsf{X}, \mathsf{F}, \mathsf{U}, \mathsf{G}, \mathsf{R}$ are called *next*, *eventually*, *until*, *globally*, and *release*, respectively. Let us note that $\mathsf{F}\varphi$ can be equivalently defined as an abbreviation for $\top \mathsf{U} \varphi$.

We define the semantics of LTL in terms of languages over infinite words. An *alphabet* is a finite set $\Sigma \subseteq \Lambda$. A *word* over alphabet Σ is an infinite sequence $w = w(0)w(1)w(2)\ldots \in \Sigma^\omega$ of letters from Σ. For every $i \in \mathbb{N}_0$, by w_i we denote the suffix of w of the form $w(i)w(i+1)w(i+2)\ldots$.

The *validity* of an LTL formula φ for $w \in \Sigma^\omega$ is defined as follows:

$$
\begin{array}{lll}
w \models \top & & \\
w \models a & \text{iff} & a = w(0) \\
w \models \neg\varphi & \text{iff} & w \not\models \varphi \\
w \models \varphi_1 \wedge \varphi_2 & \text{iff} & w \models \varphi_1 \wedge w \models \varphi_2 \\
w \models \mathsf{X}\varphi & \text{iff} & w_1 \models \varphi \\
w \models \mathsf{F}\varphi & \text{iff} & \exists i \in \mathbb{N}_0 : w_i \models \varphi \\
w \models \varphi_1 \mathsf{U} \varphi_2 & \text{iff} & \exists i \in \mathbb{N}_0 : w_i \models \varphi_2 \wedge \forall\, 0 \leq j < i : w_j \models \varphi_1
\end{array}
$$

Given an alphabet Σ, an LTL formula φ defines the language $L^\Sigma(\varphi) = \{w \in \Sigma^\omega \mid w \models \varphi\}$.

Now we define a notation for LTL fragments given by bounds on nesting depths of temporal operators. Let $O \in \{\mathsf{X}, \mathsf{F}, \mathsf{U}\}$ be a temporal operator. The nesting depth of O in a formula φ, written $O\text{-}depth(\varphi)$, is defined in the following way, where Z and Z' range over unary and binary (temporal as well as boolean) operators respectively.

$$
\begin{array}{ll}
O\text{-}depth(\top) & = 0 \\
O\text{-}depth(a) & = 0 \\
O\text{-}depth(Z\varphi) & = \begin{cases} O\text{-}depth(\varphi) + 1 & \text{if } Z = O \\ O\text{-}depth(\varphi) & \text{otherwise} \end{cases} \\
O\text{-}depth(\varphi_1 Z' \varphi_2) & = \begin{cases} \max\{O\text{-}depth(\varphi_1), O\text{-}depth(\varphi_2)\} + 1 & \text{if } Z' = O \\ \max\{O\text{-}depth(\varphi_1), O\text{-}depth(\varphi_2)\} & \text{otherwise} \end{cases}
\end{array}
$$

For all $m, n, k \in \mathbb{N}_0 \cup \{\infty\}$, we set

$$\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n, \mathsf{F}^k) = \{\varphi \mid \mathsf{U}\text{-}depth(\varphi) \leq m,\ \mathsf{X}\text{-}depth(\varphi) \leq n,\ \mathsf{F}\text{-}depth(\varphi) \leq k\}.$$

We abuse this fragment notation by omitting the upper indices equal to ∞. Moreover, we usually omit the whole operator if its index is 0. For example, by $\mathrm{LTL}(\mathsf{X}^n, \mathsf{F})$ we mean the fragment $\mathrm{LTL}(\mathsf{U}^0, \mathsf{X}^n, \mathsf{F}^\infty)$.

2.2 Alternating 1-weak Büchi automata (A1W)

The transition function of an alternating automaton assigns to each state and letter a positive boolean formula over states. The set of *positive boolean formulae* over set Q (denoted $\mathcal{B}^+(Q)$) consists of formulae $\top$ (true), $\bot$ (false), all elements of Q, and boolean combinations over Q built with $\wedge$ and $\vee$. A subset S of Q is a *model* of $\varphi \in \mathcal{B}^+(Q)$ iff φ is satisfied by the valuation assigning true just to

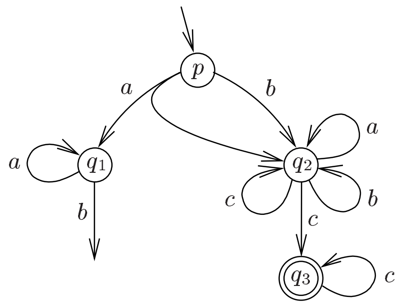

Fig. 1. The automaton accepting the language $a^*b\{a,b,c\}^*c^\omega$.

states in S. A set S is a *minimal model* of φ (denoted $S \models \varphi$) iff S is a model of φ and no proper subset of S is a model of φ.

An *alternating Büchi automaton* is a tuple $A = (\Sigma, Q, q_0, \delta, F)$, where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function, and $F \subseteq Q$ is a set of accepting states. By $A(p)$ we denote the automaton A with initial state $p \in Q$ instead of q_0.

A run of an alternating automaton is a (potentially infinite) tree. A *tree* is a set $T \subseteq \mathbb{N}_0^*$ such that if $xc \in T$, where $x \in \mathbb{N}_0^*$ and $c \in \mathbb{N}_0$, then also $x \in T$ and $xc' \in T$ for all $0 \leq c' < c$. A *Q-labeled tree* is a pair (T, r) where T is a tree and $r : T \to Q$ is a labeling function. A *run* of an automaton $A = (\Sigma, Q, q_0, \delta, F)$ over word $w \in \Sigma^\omega$ is a Q-labeled tree (T, r) such that $r(\varepsilon) = q_0$ and for each $x \in T$ the set $S = \{r(xc) \mid c \in \mathbb{N}_0, xc \in T\}$ satisfies $S \models \delta(r(x), w(|x|))$. A run (T, r) is *accepting* iff for each infinite path π in T it holds that $\mathit{Inf}(\pi) \cap F \neq \emptyset$, where $\mathit{Inf}(\pi)$ is the set of all labels (i.e. states) appearing infinitely often on π. An automaton A *accepts* a word $w \in \Sigma^\omega$ iff there exists an accepting run of A over w. A language of all words accepted by an automaton A is denoted by $L(A)$.

Let $\mathit{Succ}(p)$ denote the set $\mathit{Succ}(p) = \{q \mid \exists a \in \Sigma, S \subseteq Q : S \cup \{q\} \models \delta(p, a)\}$ of all possible successors of p, and $\mathit{Succ}'(p) = \mathit{Succ}(p) \smallsetminus \{p\}$. An automaton is called *1-weak* if there exists an ordering $<$ on the set of states Q such that $q \in \mathit{Succ}'(p)$ implies $q < p$. In the following we use *A1W automaton* or simply *automaton* meaning 'alternating 1-weak Büchi automaton'. Further, instead of $S \models \delta(a, p)$ we write $p \stackrel{a}{\to} S$ and say that an automaton has a transition leading from p to S under a. A state p of an automaton has a *loop* whenever $p \in \mathit{Succ}(p)$.

An A1W automaton $A = (\Sigma, Q, q_0, \delta, F)$ can be drawn as a graph; nodes are the states and every transition $p \stackrel{a}{\to} S$ is depicted as a branching edge labelled with a and leading from node p to the nodes in S. Edges that are not leading to any node correspond to the cases when S is the empty set. Initial and accepting states are indicated in the standard way. For example, Figure 1 depicts an automaton accepting the language $a^*b\{a,b,c\}^*c^\omega$.

2.3 LTL→A1W translation [1, 11]

In this subsection we treat every (sub)formula of the form $\mathsf{F}\varphi$ as an abbreviation for $\top \mathbin{\mathsf{U}} \varphi$.

Let φ be an LTL formula and Σ be an alphabet. The formula can be translated into an automaton A satisfying $L(A) = L^{\Sigma}(\varphi)$, where $A = (\Sigma, Q, q_\varphi, \delta, F)$ and

- the states $Q = \{q_\psi, q_{\neg\psi} \mid \psi \text{ is a subformula of } \varphi\}$ correspond to the subformulae of φ and their negations,
- the transition function δ is defined inductively as:

$$\begin{aligned}
\delta(q_\top, a) &= \top \\
\delta(q_a, b) &= \top \text{ if } a = b,\ \delta(q_a, b) = \bot \text{ otherwise} \\
\delta(q_{\neg\psi}, a) &= \overline{\delta(q_\psi, a)} \\
\delta(q_{\psi\wedge\rho}, a) &= \delta(q_\psi, a) \wedge \delta(q_\rho, a) \\
\delta(q_{\mathsf{X}\psi}, a) &= q_\psi \\
\delta(q_{\psi\mathsf{U}\rho}, a) &= \delta(q_\rho, a) \vee (\delta(q_\psi, a) \wedge q_{\psi\mathsf{U}\rho})
\end{aligned}$$

  where $\overline{\alpha}$ denotes the positive boolean formula dual to α defined by induction on the structure of α as:

$$\begin{array}{lll}
\overline{\top} = \bot & \overline{q_{\neg\psi}} = q_\psi & \overline{\beta \wedge \gamma} = \overline{\beta} \wedge \overline{\gamma} \\
\overline{\bot} = \top & \overline{q_\psi} = q_{\neg\psi} & \overline{\beta \vee \gamma} = \overline{\beta} \vee \overline{\gamma}
\end{array}$$

- the set of accepting states is $F = \{q_{\neg(\psi\mathsf{U}\rho)} \mid \psi \mathbin{\mathsf{U}} \rho \text{ is a subformula of } \varphi\}$.

We use the notation $A^{\Sigma}(\varphi)$ for the automaton given by the translation of an LTL formula φ with respect to an alphabet Σ.

For example, the translation applied on the formula $\varphi = (a \mathbin{\mathsf{U}} b) \wedge \mathsf{FG}c$ and the alphabet $\Sigma = \{a, b, c\}$ produces the automaton depicted on Figure 1, where p, q_1, q_2, q_3 stand for $q_\varphi, q_{a\mathsf{U}b}, q_{\mathsf{FG}c}, q_{\mathsf{G}c}$, respectively.

2.4 A1W→LTL translation [2, 3]

Let $A = (\Sigma, Q, q_0, \delta, F)$ be an A1W automaton. For each $p \in Q$ we define an LTL formula φ_p such that $L^{\Sigma}(\varphi_p) = L(A(p))$ (in particular $L^{\Sigma}(\varphi_{q_0}) = L(A)$). The definition proceeds by induction respecting the ordering of states; the formula φ_p employs formulae of the form φ_q where $q \in \mathit{Succ}'(p)$. This is the point where the 1-weakness of the automaton is used. To illustrate the inductive step of the translation, let us consider the situation depicted on Figure 2. The formula corresponding to state p is $\varphi_p = (a \wedge \mathsf{X}\varphi_q) \mathbin{\mathsf{U}} (b \wedge \mathsf{X}\varphi_r)$.

Before we give a formal definition of φ_p, we introduce some auxiliary formulae. Let $a \in \Sigma$ be a letter and $S \subseteq Q$ be a set of states.

$$\theta(a, S) = a \wedge \bigwedge_{q \in S} \mathsf{X}\varphi_q \qquad \alpha_p = \bigvee_{\substack{p \xrightarrow{a} S \\ p \in S}} \theta(a, S \smallsetminus \{p\}) \qquad \beta_p = \bigvee_{\substack{p \xrightarrow{a} S \\ p \notin S}} \theta(a, S)$$

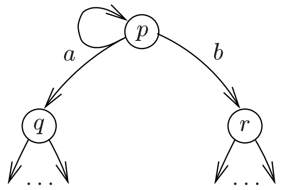

Fig. 2. Part of an automaton translated into the formula $\varphi_p = (a \wedge \mathsf{X}\varphi_q) \,\mathsf{U}\, (b \wedge \mathsf{X}\varphi_r)$.

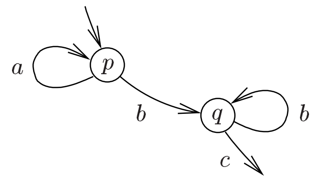

Fig. 3. An automaton for the formula $a \,\mathsf{U}\, (b \wedge (b \,\mathsf{U}\, c))$ and alphabet $\{a, b, c\}$.

The formula $\theta(a, S)$ represent a situation where the automaton makes a transition under a into the set of states S. Formulae α_p and β_p correspond to all transitions leading from state p; α_p covers transitions with a loop while β_p covers the others. The definition of φ_p then depends on whether p is an accepting state or not.

$$\varphi_p = \begin{cases} \alpha_p \,\mathsf{U}\, \beta_p & \text{if } p \notin F \\ (\alpha_p \,\mathsf{U}\, \beta_p) \vee \mathsf{G}\alpha_p & \text{if } p \in F \end{cases}$$

Given an automaton A with an initial state q_0, we set $\varphi(A) = \varphi_{q_0}$.

3 Improved A1W→LTL translation

The weak point of the A1W→LTL translation presented above is that for each successor $q \in \mathit{Succ}'(p)$ of a state p the formula φ_p contains a subformula $\mathsf{X}\varphi_q$ even if the X operator is not needed. This is illustrated by the automaton A in Figure 3 produced by translating formula $a \,\mathsf{U}\, (b \wedge (b \,\mathsf{U}\, c))$ with respect to alphabet $\{a, b, c\}$. The reverse translation provides an equivalent formula $\varphi(A) = a \,\mathsf{U}\, (b \wedge \mathsf{X}(b \,\mathsf{U}\, c))$.

Let $p \xrightarrow{a} S$ be a transition and $X \subseteq S$. We now formulate conditions that are sufficient to omit the X operator in front of φ_q (for every $q \in X$) in the subformula of φ_p corresponding to the transition $p \xrightarrow{a} S$.

Definition 1. *Let $p \xrightarrow{a} S$ be a transition of an automaton A. A set $X \subseteq S \smallsetminus \{p\}$ is said to be* X-free for $p \xrightarrow{a} S$ *if the following conditions hold.*

1. *For each $q \in X$ there is $S'_q \subseteq S$ such that $q \xrightarrow{a} S'_q$.*

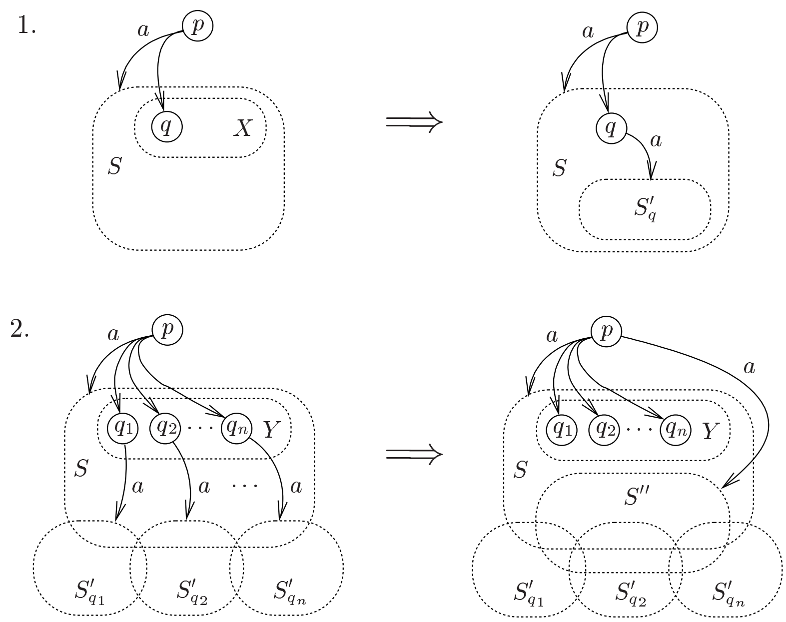

Fig. 4. The conditions for X-freeness.

2. *Let $Y \subseteq X$ and for each $q \in Y$ let $S'_q \subseteq Q$ be a set satisfying $q \xrightarrow{a} S'_q$ and $q \notin S'_q$. Then there exists a set $S'' \subseteq (S \smallsetminus Y) \cup \bigcup_{q \in Y} S'_q$ satisfying $p \xrightarrow{a} S''$.*

Figure 4 illustrates the conditions for X-freeness. Please note that it can be the case that $p \in S$. Further, in the first condition it can be the case that $q \in S'_q$.

It is easy to see that the empty set is X-free for every transition. Further, every subset of an X-free set for a transition is X-free for the transition as well. On the other hand, Figure 5 demonstrates that a union of two X-free sets need not be X-free.

Let Xfree be an arbitrary but fixed function assigning to each transition $p \xrightarrow{a} S$ a set that is X-free for $p \xrightarrow{a} S$. We now introduce an improved A1W→LTL translation. Roughly speaking, the translation omits the X operators in front of subformulae which correspond to the states in X-free sets given by the function Xfree.

The improved A1W→LTL translation exhibits similar structure as the original one. Instead of formulae of the form $\theta(a, S)$ representing a transition under a leading from an arbitrary state p to S, we define a specialized formula $\theta'_p(a, S)$ for each transition $p \xrightarrow{a} S$.

$$\theta'_p(a, S) = a \ \wedge \bigwedge_{\substack{q \in S \smallsetminus \mathsf{Xfree}(p \xrightarrow{a} S) \\ q \neq p}} \mathsf{X}\varphi'_q \ \wedge \bigwedge_{q \in \mathsf{Xfree}(p \xrightarrow{a} S)} \varphi'_q$$

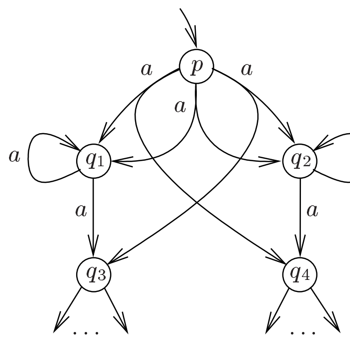

Fig. 5. The sets $\{q_1\}, \{q_2\}$ are X-free for $p \xrightarrow{a} \{q_1, q_2\}$ while the set $\{q_1, q_2\}$ is not.

$$\alpha'_p = \bigvee_{\substack{p \xrightarrow{a} S \\ p \in S}} \theta'_p(a,S) \qquad\qquad \beta'_p = \bigvee_{\substack{p \xrightarrow{a} S \\ p \notin S}} \theta'_p(a,S)$$

In the following definition of a formula φ'_p we identify some cases when U can be replaced by "weaker" operators F or G. To this end we define two special types of states. A state p is of the F-*type* if there is a transition $p \xrightarrow{a} \{p\}$ for every $a \in \Sigma$. A state p is of the G-*type* if every transition of the form $p \xrightarrow{a} S$ satisfies $p \in S$.

$$\varphi'_p = \begin{cases} \beta'_p & \text{if } p \notin \mathit{Succ}(p) \\ \bot & \text{if } p \in \mathit{Succ}(p),\ p \notin F,\ p \text{ is of } \mathsf{G}\text{-type} \\ \mathsf{F}\beta'_p & \text{if } p \in \mathit{Succ}(p),\ p \notin F,\ p \text{ is of } \mathsf{F}\text{-type and not of } \mathsf{G}\text{-type} \\ \alpha'_p \,\mathsf{U}\, \beta'_p & \text{if } p \in \mathit{Succ}(p),\ p \notin F,\ p \text{ is neither of } \mathsf{F}\text{-type nor of } \mathsf{G}\text{-type} \\ \top & \text{if } p \in \mathit{Succ}(p),\ p \in F,\ p \text{ is of } \mathsf{F}\text{-type} \\ \mathsf{G}\alpha'_p & \text{if } p \in \mathit{Succ}(p),\ p \in F,\ p \text{ is of } \mathsf{G}\text{-type and not of } \mathsf{F}\text{-type} \\ (\alpha'_p \,\mathsf{U}\, \beta'_p) \vee \mathsf{G}\alpha'_p & \text{if } p \in \mathit{Succ}(p),\ p \in F,\ p \text{ is neither of } \mathsf{F}\text{-type nor of } \mathsf{G}\text{-type} \end{cases}$$

By $\varphi^{\mathsf{Xfree}}(A)$ we denote the formula φ'_{q_0} given by the improved translation using the function Xfree. The following Theorems 1 and 2 say that the translation is correct and that it does not waste temporal operators.

Theorem 1. *Let A be an A1W automaton over alphabet Σ. Let* Xfree *be a function assigning an* X*-free set to each transition of A. Then $L(A) = L^{\Sigma}(\varphi^{\mathsf{Xfree}}(A))$.*

Theorem 1 is proved by induction with respect to ordering of states in the automaton A. The theorem is a direct corollary of the following lemma.

Lemma 1. *Let $p \xrightarrow{a} S$ be a transition of an A1W automaton A such that for each $q \in \mathit{Succ}'(p)$ the equivalence $\varphi_q \iff \varphi'_q$ holds. Then $\theta(a, S \smallsetminus \{p\}) \implies \theta'_p(a,S)$. Further, $\theta'_p(a,S) \implies \beta_p \vee \alpha_p$. Moreover, if $p \notin S$ then $\theta'_p(a,S) \implies \beta_p$.*

Theorem 2. *For each formula $\varphi \in \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n, \mathsf{F}^k)$ and each alphabet Σ there exists a function* Xfree *such that* $\varphi^{\mathsf{Xfree}}(A^{\Sigma}(\varphi)) \in \mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n, \mathsf{F}^k)$.

The function Xfree can be effectively constructed from the transition relation of the automaton. For further details about the construction and for full proofs see [19].

The improved translation enables us to study relations between fragments of the form $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n, \mathsf{F}^k)$ and classes of A1W automata. In particular, we can provide alternative definitions of language classes corresponding to some previously studied LTL fragments, namely fragments of the form $\mathrm{LTL}(\mathsf{U}^k, \mathsf{X}, \mathsf{F})$ constituting the so-called *until hierarchy* [12, 13], fragments of the form $\mathrm{LTL}(\mathsf{U}, \mathsf{X}^n)$, $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X})$, or $\mathrm{LTL}(\mathsf{U}^m, \mathsf{X}^n)$ studied in [14, 15], and the fragments $\mathrm{LTL}(\mathsf{X}, \mathsf{F})$ also called *restricted LTL* [16]. Due to the lack of space we mention only the alternative definition of languages definable in $\mathrm{LTL}(\mathsf{X}, \mathsf{F})$. The other cases are a bit more complicated and can be found in [19].

Lemma 2. *A language is definable by a formula of* $\mathrm{LTL}(\mathsf{X}, \mathsf{F})$ *if and only if there exists an A1W automaton recognizing the language such that every state with a loop is of* F*-type or* G*-type.*

4 Until-release hierarchy and A1W automata

The *until-release hierarchy* of LTL formulae has been introduced in [4]. It is based on alternation depth of U and R operators. Therefore it is also called *alternating hierarchy*. This hierarchy has a strong connection to the *hierarchy of temporal properties* introduced by Manna and Pnueli [17, 18]. Moreover, there is a relation between classes of until-release hierarchy and complexity of their model checking problem (see [4]).

Definition 2. *The classes* $\mathsf{UR}_i, \mathsf{RU}_i$ *of the Until-Release hierarchy are defined inductively.*

- *The classes* UR_0 *and* RU_0 *are both identical to* $\mathrm{LTL}(\mathsf{X})$.
- *The class* UR_{i+1} *is the least set containing* RU_i *and closed under the application of operators* $\wedge, \vee, \mathsf{X}$, *and* U.
- *The class* RU_{i+1} *is the least set containing* UR_i *and closed under the application of operators* $\wedge, \vee, \mathsf{X}$, *and* R.

Let us note that the hierarchy collapses on the third level with respect to its expressive power. More precisely, each language is definable by LTL if and only if it is definable by a positive boolean combination of UR_2 and RU_2 formulae. These formulae are contained in UR_3 as well as in RU_3. In the following we identify a fragment of the alternating hierarchy with the set of languages defined by formulae of this fragment.

We now define the alternation depth of nonaccepting and accepting states in the graph of an A1W automaton. We also define classes of languages recognized by automata with a given alternation depth.

Definition 3. *Let $A = (\Sigma, Q, q_0, \delta, F)$ be an A1W automaton. For each $i \in \mathbb{N}_0$ we inductively define sets of states σ_i and π_i as follows.*

- σ_0 *is the smallest set of states satisfying*
  - $\{p \mid p \notin F \text{ and } Succ(p) = \emptyset\} \subseteq \sigma_0$ *and*
  - *if* $p \notin F$ *and* $Succ(p) \subseteq \sigma_0$ *then* $p \in \sigma_0$,
- π_0 *is the smallest set of states satisfying*
  - $\{p \mid p \in F \text{ and } Succ(p) = \emptyset\} \subseteq \pi_0$ *and*
  - *if* $p \in F$ *and* $Succ(p) \subseteq \pi_0$ *then* $p \in \pi_0$,
- σ_{i+1} *is the smallest set of states satisfying*
  - $\sigma_i \cup \pi_i \subseteq \sigma_{i+1}$ *and*
  - *if* $p \notin F$ *and* $Succ'(p) \subseteq \sigma_{i+1}$ *then* $p \in \sigma_{i+1}$,
- π_{i+1} *is the smallest set of states satisfying*
  - $\sigma_i \cup \pi_i \subseteq \pi_{i+1}$ *and*
  - *if* $p \in F$ *and* $Succ'(p) \subseteq \pi_{i+1}$ *then* $p \in \pi_{i+1}$.

We also define functions $\sigma_A, \pi_A : Q \longrightarrow \mathbb{N}_0$ *as*

$$\sigma_A(p) = min\{i \mid p \in \sigma_i\} \quad \text{and} \quad \pi_A(p) = min\{i \mid p \in \pi_i\}.$$

Finally, for each $i \in \mathbb{N}_0$ *we define sets* Σ_i *and* Π_i *as*

$$\Sigma_i = \{L(A) \mid A = (\Sigma, Q, q_0, \delta, F) \text{ is an A1W automaton and } \sigma_A(q_0) \leq i\},$$
$$\Pi_i = \{L(A) \mid A = (\Sigma, Q, q_0, \delta, F) \text{ is an A1W automaton and } \pi_A(q_0) \leq i\}.$$

The following theorem says that a language is definable by a formula of UR_i if and only if it is recognized by an A1W automaton with alternation depth of nonaccepting and accepting states at most i. An analogous statement holds for RU_i and alternation depth of accepting and nonaccepting states. It is worth mentioning that the proof of the following theorem is not as simple as one can think when looking at the definition of a formula φ_p in the standard A1W→LTL translation. See [19] for details.

Theorem 3. *For each* $i \in \mathbb{N}_0$ *it holds that* $\mathsf{UR}_i = \Sigma_i$ *and* $\mathsf{RU}_i = \Pi_i$.

The theorem allows us to transform the results proved for the until-release hierarchy in [4] into statements about our hierarchy of Σ_i and Π_i classes. This is exemplified by the two following corollaries. For definitions of language classes mentioned in the latter corollary (safety, guarantee, obligation,...) we refer to [17, 18].

Corollary 1. *The hierarchy of* Σ_i *and* Π_i *classes collapses on the third level, i.e.* $\Sigma_3 = \Pi_3 = \Sigma_i = \Pi_i$ *for all* $i > 3$.

Corollary 2. *A language definable in LTL is in safety, guarantee, obligation, response, persistence, or reactivity class iff it is in* Π_1, Σ_1, $\Pi_2 \cap \Sigma_2$, Π_2, Σ_2, *or* $\Pi_3 \cap \Sigma_3$, *respectively.*

5 Summary and future work

The paper presents two main results. The first is the improved translation of A1W automata into LTL formulae that are language equivalent. The second result is a new automata-based definition of classes in the until-release hierarchy [4]. We also provide some corollaries of these results and indicate further applications.

Besides the presented results our research brought several topics for future work. For example, we would like to know whether there are some more general or/and simpler conditions for a set to be X-free (see Definition 1). Another interesting question is the relation between the sizes of LTL formulae and equivalent A1W automata. Both standard and improved A1W$\rightarrow$LTL translations can be modified to produce formulae that can be represented by directed acyclic graphs of linear size with respect to the size of the original automata. However, we conjecture that A1W automata can be exponentially more succinct than LTL if we stick with the standard representation of LTL formulae.

References

1. Muller, D.E., Saoudi, A., Schupp, P.E.: Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In: Proceedings of the 3rd IEEE Symposium on Logic in Computer Science (LICS 1988), IEEE Computer Society Press (1988) 422–427
2. Rohde, S.: Alternating automata and the temporal logic of ordinals. PhD thesis, University of Illinois at Urbana-Champaign (1997)
3. Löding, C., Thomas, W.: Alternating automata and logics over infinite words (extended abstract). In van Leeuwen, J., et al., eds.: Theoretical computer science: exploring new frontiers of theoretical informatics: International Conference IFIP TCS 2000. Volume 1872 of Lecture Notes in Computer Science., Springer-Verlag (2000) 521–535
4. Černá, I., Pelánek, R.: Relating hierarchy of temporal properties to model checking. In: Mathematical Foundations of Computer Science (MFCS). Volume 2747 of Lecture Notes in Computer Science., Springer (2003)
5. Wolper, P., Vardi, M.Y., Sistla, A.P.: Reasoning about infinite computation paths (extended abstract). In: 24th Annual Symposium on Foundations of Computer Science, IEEE (1983) 185–194
6. Vardi, M.Y., Wolper, P.: Reasoning about infinite computations. Information and Computation **115** (1994) 1–37
7. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification. In: Proceedings of the First Symposium on Logic in Computer Science, Cambridge (1986) 322–331
8. Wolper, P.: Temporal logic can be more expressive. Information and Control **56** (1983) 72–99
9. Gastin, P., Oddoux, D.: Fast LTL to Büchi automata translation. In Berry, G., Comon, H., Finkel, A., eds.: Proceedings of the 13th Conference on Computer Aided Verification (CAV'01). Volume 2102 of Lecture Notes in Computer Science., Springer (2001) 53–65

10. Tauriainen, H.: On translating linear temporal logic into alternating and nondeterministic automata. Research Report A83, Helsinki University of Technology, Laboratory for Theoretical Computer Science (2003)
11. Vardi, M.Y.: Alternating automata: Unifying truth and validity checking for temporal logics. In McCune, W., ed.: Proceedings of the 14th International Conference on Automated Deduction. Volume 1249 of LNAI., Springer (1997) 191–206
12. Thérien, D., Wilke, T.: Temporal logic and semidirect products: An effective characterization of the until hierarchy. In: 37th Annual Symposium on Foundations of Computer Science (FOCS '96), IEEE (1996) 256–263
13. Etessami, K., Wilke, T.: An until hierarchy and other applications of an Ehrenfeucht-Fraïssé game for temporal logic. Information and Computation **160** (2000) 88–108
14. Kučera, A., Strejček, J.: The stuttering principle revisited. Acta Informatica (2005) To appear.
15. Kučera, A., Strejček, J.: Characteristic patterns for LTL. In: Proceedings of SOFSEM 2005. Volume 3381 of Lecture Notes in Computer Science., Springer-Verlag (2005) 239–249
16. Perrin, D., Pin, J.E.: Infinite words. Volume 141 of Pure and Applied Mathematics. Elsevier (2004)
17. Manna, Z., Pnueli, A.: A hierarchy of temporal properties. In: Proc. ACM Symposium on Principles of Distributed Computing, ACM Press (1990) 377–410
18. Chang, E., Manna, Z., Pnueli, A.: Characterization of temporal property classes. In Kuich, W., ed.: Automata, Languages and Programming, 19th International Colloquium (ICALP '92). Volume 623 of Lecture Notes in Computer Science., Springer-Verlag (1992) 474–486
19. Pelánek, R., Strejček, J.: Deeper connections between ltl and alternating automata. Technical Report FIMU-RS-2004-08, Faculty of Informatics, Masaryk University in Brno (2004) Available at `http://www.fi.muni.cz/reports/`.