# Automata and temporal logic over arbitrary linear time

Julien Cristau

# Automata and temporal logic over arbitrary linear time

**Julien Cristau**
LIAFA — CNRS & Université Paris 7

ABSTRACT. Linear temporal logic was introduced in order to reason about reactive systems. It is often considered with respect to infinite words, to specify the behaviour of long-running systems. One can consider more general models for linear time, using words indexed by arbitrary linear orderings. We investigate the connections between temporal logic and automata on linear orderings, as introduced by Bruyère and Carton. We provide a doubly exponential procedure to compute from any LTL formula with Until, Since, and the Stavi connectives an automaton that decides whether that formula holds on the input word. In particular, since the emptiness problem for these automata is decidable, this transformation gives a decision procedure for the satisfiability of the logic.

## 1 Introduction

Temporal logic, in particular LTL, was proposed by Pnueli to specify the behaviour of reactive systems [12]. The model of time usually considered is the ordered set of natural numbers, and executions of the system are seen as infinite words on some set of atomic propositions. This logic was shown to have the same expressive power as the first order logic of order [11], but it provides a more convenient formalism to express verification properties. It is also more tractable: while the satisfiability problem of FO is non-elementary [18], it was shown in [17] that the decision problem of LTL with Until and Since on $\omega$-words is PSPACE-complete. This logic has also strong ties with automata, with important work to provide efficient translations to Büchi automata, e.g. [10].

Within this time model, a number of extensions of the logic and the automata model have been studied. But one can also consider more general models of time: general linear time could be useful in different settings, including concurrency, asynchronous communication, and others, where using the set of integers can be too simplistic. Possible choices include ordinals, the reals, or even arbitrary linear orderings. In terms of expressivity, while LTL with Until and Since is expressively complete (*i.e.* equivalent to FO) on Dedekind-complete orderings (which includes the ordering of the reals as well as all ordinals), this does not hold in the general case. Two more connectives, the future and past Stavi operators, are necessary to handle gaps [9] when considering arbitrary linear orderings.

Over ordinals, LTL with Until and Since has been shown to have a PSPACE-complete satisfiability problem [7]. Over the ordering of the real numbers, satisfiability of LTL with until and since is PSPACE-complete, but satisfiability of MSO is undecidable. Over general linear time, first order logic has been shown to be decidable, as well as universal monadic second order logic. Reynolds shows in [13] that the satisfiability problem of temporal logic with only the Until connective is also PSPACE-complete, and conjectures that this might stay true when adding the Since connective. The upper bound in [7] is obtained by reducing the satisfiability of LTL formulae to the accessibility problem in an appropriate automata model, accepting words indexed by ordinals. In this paper, we focus on the general case of arbitrary linear orderings, using the full logic with Until, Since and both Stavi connectives. Our aim is to investigate the connections between LTL and automata in this setting.

Automata on linear orderings were introduced by Bruyère and Carton [3]. This model extends traditional finite automata using "limit" transitions to handle positions with no successor or predecessor, furthering Büchi's model of automata on words of ordinal length [4]. Carton showed in [5] that accessibility over scattered ordering is decidable in polynomial time, and in [14] it was shown that these automata can be complemented over countable scattered linear orderings. The accessibility result can be extended to arbitrary orderings [6].

From any formula in this logic, we define an automaton which determines whether the formula holds on its input word. Satisfiability of the formula is reduced to accessibility in this automaton, and that way we get decidability of the satisfiability problem of LTL with Until, Since and the Stavi operators for any rational subclass.

Section 2 presents some definitions about linear orderings, linear temporal logic, and the model of automata used. Section 3 introduces our main result, an algorithm to translate any LTL formula into a corresponding automaton. Section 4 discusses the expressivity of the logic and automata considered, and looks at some natural fragments.

# 2 Definitions

## 2.1 Linear orderings

We first recall some basic definitions about orderings, and introduce some notations. For a complete introduction to linear orderings, the reader is referred to [15]. A *linear ordering* $J$ is a totally ordered set $(J, <)$ (considered modulo isomorphism). The sets of integers ($\omega$), of rational numbers ($\eta$), and of real numbers with the usual orderings are all linear orderings.

Let $J$ and $K$ be two linear orderings. One defines the reversed ordering $-J$ as the ordering obtained by reversing the relation $<$ in $J$, and the ordering $J + K$ as the disjoint union $J \sqcup K$ extended with $j < k$ for any $j \in J$ and $k \in K$. For example, $-\omega$ is the ordering of negative integers. $-\omega + \omega$ is the usual ordering of $\mathbb{Z}$, also denoted by $\zeta$.

A non-empty subset $K$ of an ordering $J$ is an *interval* if for any $i < j < k$ in $J$, if $i \in K$ and $k \in K$ then $j \in K$. In order to define the runs of an automaton, we use the notion of cut. A *cut* of an ordering $J$ is a partition $(K, L)$ of $J$ such that for any $k \in K$ and $l \in L$, $k < l$. We denote by $\hat{J}$ the set of cuts of $J$. This set is equipped with the order defined by $(K_1, L_1) < (K_2, L_2)$ if $K_1 \subsetneq K_2$. This ordering can be extended to $J \cup \hat{J}$ in a natural way ($(K, L) < j$ iff $j \in L$). Notice that $\hat{J}$ always has a smallest and a biggest element, respectively $c_{\min} = (\emptyset, J)$ and $c_{\max} = (J, \emptyset)$. For example, the set of cuts of the finite ordering $\{0, 1, \ldots, n-1\}$ is the ordering $\{0, 1, \ldots, n\}$, and the set of cuts of $\omega$ is $\omega + 1$.

For any element $j$ of $J$, there are two successive cuts $c_j^-$ and $c_j^+$, respectively $(\{i \in J \mid i < j\}, \{i \in J \mid j \leq i\})$ and $(\{i \in J \mid i \leq j\}, \{i \in J \mid j < i\})$. A *gap* in an ordering $J$ is a cut $c$ which is not an extremity ($c_{\max}$ or $c_{\min}$), and has neither a successor nor a predecessor.

Given an alphabet $\Sigma$, a *word* of length $J$ is a sequence $(a_j)_{j \in J}$ of elements of $\Sigma$ indexed by $J$. For example, $(ab)^\omega$ is a word of length $\omega$; the sequence $ab^\omega ab^\omega a$ is a word of length $\omega + \omega + 1$, and $(ab^\omega)^\omega$ is a word of length $\omega^2$.

### 2.2 Temporal logic

We use words over linear orderings to model the behaviour of systems over linear time. To express properties of these systems, we consider linear temporal logic. The set of LTL formulae is defined by the following grammar, where $p$ ranges over a set AP of atomic propositions: $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\mathcal{U}\varphi \mid \varphi\mathcal{S}\varphi \mid \varphi\mathcal{U}'\varphi \mid \varphi\mathcal{S}'\varphi$

Besides the usual boolean operators, we have four temporal connectives. The $\mathcal{U}$ connective is called "Until", and $\mathcal{S}$ is called "Since". $\mathcal{U}'$ and $\mathcal{S}'$ are respectively the future and past Stavi connectives. Other usual connectives such as "Next" ($\mathcal{X}$), "Eventually" ($\mathcal{F}$), "Always" ($\mathcal{G}$) can be defined using these, as we see below.

These formulae are interpreted on words over the alphabet $2^{\mathrm{AP}}$. A letter in those words is the set of atomic propositions that hold at the corresponding position. Let $x = (x_j)_{j\in J}$ a word of length $J$. A formula $\varphi$ is evaluated at a particular position $i$ in $x$; we say that $\varphi$ holds at position $i$ in $x$, and we write $x, i \models \varphi$, using the following semantics:

$$
\begin{array}{rcl}
x,i \models p & \text{if} & p \in x_i \\
x,i \models \neg\psi & \text{if} & x,i \not\models \psi \\
x,i \models \psi_1 \vee \psi_2 & \text{if} & x,i \models \psi_1 \text{ or } x,i \models \psi_2 \\
x,i \models \psi_1\mathcal{U}\psi_2 & \text{if} & \text{there exists } j > i \text{ such that } x,j \models \psi_2, \\
 & & \text{and for any } k \text{ such that } i < k < j, \text{ we have } x,k \models \psi_1 \\
x,i \models \psi_1\mathcal{S}\psi_2 & \text{if} & -x,i \models \psi_1\mathcal{U}\psi_2 \text{ where } -x \text{ is the reversed word } (a_j)_{j\in -J} \\
x,i \models \psi_1\mathcal{U}'\psi_2 & \text{if} & \text{there exists a gap } c \in \hat{J} \text{ verifying three properties:} \\
 & (1) & x,j \models \psi_1 \text{ for any position } j \text{ such that } i < j < c \\
 & (2) & \text{there is no interval starting at } c \text{ where } \psi_1 \text{ is always true} \\
 & & (\textit{i.e.}\ \forall c < k\ \exists c < j < k\ x,j \models \neg\psi_1), \text{ and} \\
 & (3) & \psi_2 \text{ is always true in some interval starting at } c \\
x,i \models \psi_1\mathcal{S}'\psi_2 & \text{if} & -x,i \models \psi_1\mathcal{U}'\psi_2 \text{ (it is the corresponding past connective)}
\end{array}
$$

Note that we use a "strict" semantic for the Until operator, contrary to a common definition, which would be:

$$x,i \models \psi_1\mathcal{U}^{ns}\psi_2 \quad \text{if} \quad \text{there exists } j \geq i \text{ such that } x,j \models \psi_2 \text{ and } x,k \models \psi_1 \text{ for any } i \leq k < j.$$

In the strict version, the current position $i$ is not considered for either the $\psi_1$ or the $\psi_2$ part of the definition. Using the strict or non-strict version makes no difference when considering $\omega$-words, but in the case of arbitrary orderings, the strict Until is more powerful, as noted by Reynolds in [13].

The formula "Next $\varphi$", or $\mathcal{X}\,\varphi$, is equivalent to $\bot\mathcal{U}\varphi$. "Eventually $\varphi$", noted $\mathcal{F}\,\varphi$, is $\varphi \vee (\top\mathcal{U}\varphi)$, and "always $\varphi$", noted $\mathcal{G}\,\varphi$, can be expressed as $\neg(\mathcal{F}(\neg\varphi))$.

Given a word $x$ of length $J$, the *truth word* of $\varphi$ on $x$ is the word $v_\varphi(x)$ of length $J$ over the alphabet $\{0,1\}$ where the position $j$ is labelled by 1 iff $x,j \models \varphi$. A formula is *valid* if its truth word on any input only has ones. A formula is *satisfiable* if there exists an input word such that the truth word contains a one.

Consider the formula $\varphi = \neg a \wedge (\mathcal{G} \neg \mathcal{X}\, a)$, with AP $= \{a\}$. If $x = (a\emptyset)^\omega$ (where $a$ stands for $\{a\}$), then $v_\varphi(x) = 0^\omega$ (at every position, either $a$ is true or $a$ is true in the successor). On the other hand, if $x = a\emptyset^\omega a\emptyset^\omega a$, then $v_\varphi(x) = 01^\omega 01^\omega 0$: at positions 0, $\omega$ and at the last position, $a$ is true so the formula doesn't hold; at all other positions, $a$ is false, and there is no position in the input word where $\mathcal{X}\, a$ holds.

The *satisfiability problem* for a formula $\varphi$ consists in deciding whether there exists a word $w$ and a position $i$ in $w$ such that $w, i \models \varphi$. As FO is decidable, and every LTL formula can be expressed using first order, satisfiability of LTL is decidable. Note however that in terms of complexity FO is already non-elementary on finite words [18], which is not true of LTL.

### 2.3 Automata

On infinite words, Büchi automata can be used to decide satisfiability of LTL formulae. In the case of words over linear orderings, a model of automata has been introduced in [3]. Instead of accepting or rejecting each input word, as in the case of $\omega$-words, we use these automata to compute the truth words corresponding to an LTL formula. Our model of automata thus has an output letter on each transition, so they are actually letter-to-letter transducers, which make composition easier (see Section 3.1).

An *automaton* is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, I, F)$ where $Q$ is a finite set of states, $\Sigma$ is a finite input alphabet, $\Gamma$ is a finite output alphabet, $I$ and $F$ are subsets of $Q$, respectively the set of initial and final states, and $\delta \subseteq (Q \times \Sigma \times \Gamma \times Q) \cup (2^Q \times Q) \cup (Q \times 2^Q)$ is the set of transitions. We note:

- $p \xrightarrow{a|b} q$ if $(p, a, b, q) \in \delta$ (*successor* transition)
- $P \to q$ if $(P, q) \in \delta$ (*left limit* transition)
- $q \to P$ if $(q, P) \in \delta$ (*right limit* transition).

Consider a word $x = (q_j)_{j \in J}$ over $Q$. We define the left and right limit sets of $x$ at position $j \in J$ as the sets of labels that appear arbitrarily close to $j$ (respectively to its left and to its right). Formally:

$$\lim_{j^-} x = \{q \in Q \mid \forall k < j\ \exists i\ k < i < j \wedge q_i = q\}$$
$$\lim_{j^+} x = \{q \in Q \mid \forall k > j\ \exists i\ j < i < k \wedge q_i = q\}$$

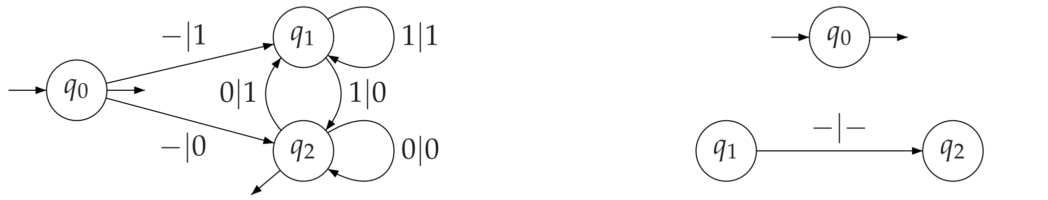
Note that $\lim_{j^-} x$ is non-empty if and only if the transition to $j$ is a left limit, and similarly for $\lim_{j^+} x$ if the transition from $j$ is a right limit. These sets help define the possible limit transitions in a run.

Given an automaton $\mathcal{A}$, an accepting run of $\mathcal{A}$ on a word $x = (x_j)_{j \in J}$ is a word $\rho$ of length $\hat{J}$ over $Q$ such that:

- $\rho_{c_{\min}} \in I$ and $\rho_{c_{\max}} \in F$;
- for each $i \in J$, there exists $y_i \in \Gamma$ such that $\rho_{c_i^-} \xrightarrow{x_i|y_i} \rho_{c_i^+}$;
- if $c \in \hat{J}$ has no predecessor, $\lim_{c^-} \rho \to \rho_c$, and if $c \in \hat{J}$ has no successor, $\rho_c \to \lim_{c^+} \rho$.

**EXAMPLE 1.** *The first automaton in Figure 1 outputs* 1 *at each position immediately followed by a* 1 *in the input word, and* 0 *at other positions.*

*The second automaton accepts input words whose length is a linear ordering without first or last element, and without two consecutive elements (i.e.* dense *orderings). The notation $P \to q_0, q_1$ means that there is a transition $P \to q_0$ and a transition $P \to q_1$.*




Limits: $P \to q_0$ $\quad q_0 \to P$ $\quad q_2 \to P$
for any $P \subseteq \{q_0, q_1, q_2\}$

Limits: $\{q_1, q_2\}, \{q_0, q_1, q_2\} \to q_0, q_1$
$q_1, q_2 \to \{q_1, q_2\}, \{q_0, q_1, q_2\}$

Figure 1: Example automata

In [5], Carton proves that the accessibility problem on these automata can be solved in polynomial time, when only considering scattered orderings. This result can be extended to arbitrary orderings [6] as it is done for rational expressions in [2]. The idea is to build an automaton over finite words which simulates the paths in the initial automaton and remembers their contents. In order to handle the general case (as opposed to only scattered orderings), the added operation is called "shuffle": $sh(w_1, \ldots, w_n) = \Pi_{j \in J} x_j$ where $J$ is a dense and complete ordering without a first or last element, partitioned in dense suborderings $J_1 \ldots J_n$, such that $x_j = w_i$ if $j \in J_i$. Looking at automata, this means that if there are paths from $p_1$ to $q_1$ with content $P_1$, ..., from $p_n$ to $q_n$ with content $P_n$, and transitions from $P_1 \cup \cdots \cup P_n$ to each $p_i$, transitions from each $q_i$ to $P_1 \cup \cdots \cup P_n$, a transition from $p$ to $P_1 \cup \cdots \cup P_n$ and a transition from $P_1 \cup \cdots \cup P_n$ to $q$, then there is a path from $p$ to $q$.

## 3 Translation between formulae and automata

Over $\omega$-words, problems on temporal logics are commonly solved using tableau methods [20], or automata-based techniques [19]. In this work we extend the correspondence between LTL and automata to words over linear orderings. Our main result is Theorem 2.

**THEOREM 2.** *For every LTL formula $\varphi$, there is an automaton $\mathcal{A}_\varphi$ which given any input word $x$ outputs the truth word $v_\varphi(x)$.*

Moreover, this automaton $\mathcal{A}_\varphi$ can be effectively computed, and has a number of states exponential in the size of $\varphi$. Because we can compute the product of $\mathcal{A}_\varphi$ with any given automaton and check for its emptiness, we get Corollary 3, which states that given a temporal formula and a rational property (*i.e.* an automaton on words over linear orderings), we can check whether there exists a model of the formula which is accepted by the automaton.

**COROLLARY 3.** *The satisfiability problem for any rational subclass is decidable.*

The idea is to build $\mathcal{A}_\varphi$ by induction on the formula. We construct an elementary automaton for each logical connective. We use composition and product operations to build inductively the automaton of any LTL formula from elementary automata. All automata used in this proof have the particular property that there exists exactly one accepting run for each possible input word, *i.e.* they are non-deterministic, but also non-ambiguous. This property is preserved by composition and product.

The structure of the proof is the following: we define the composition and product operators on automata, then we present the elementary automata that are needed to encode logical connectives. Finally, we give the inductive method to build the automaton corresponding to a formula from elementary ones.

### 3.1 Product, composition and elementary automata

Let $\mathcal{A}_1 = (Q_1, \Sigma, \Gamma, \delta_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma', \Delta, \delta_2, I_2, F_2)$ be two automata. The product consists in running both automata with the same input alphabet in parallel, and outputting the combination of their outputs. If $\mathcal{A}_1$'s output alphabet and $\mathcal{A}_2$'s input alphabet are the same, the composition consists in running $\mathcal{A}_2$ over $\mathcal{A}_1$'s output. We use the notation $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$ for the first and second projections.

**DEFINITION 4.** *Suppose that $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same input alphabet, i.e. $\Sigma = \Sigma'$. The* product *of $\mathcal{A}_1$ and $\mathcal{A}_2$ is the automaton $\mathcal{A}_1 \times \mathcal{A}_2 = (Q_1 \times Q_2, \Sigma, \Gamma \times \Delta, \delta, I_1 \times I_2, F_1 \times F_2)$, where $\delta$ contains the following transitions:*

- $(q_1, q_2) \xrightarrow{a|b,c} (q_1', q_2')$ *if* $q_1 \xrightarrow{a|b} q_1'$ *and* $q_2 \xrightarrow{a|c} q_2'$,
- $(q_1, q_2) \to P$ *if* $q_1 \to \pi_1(P)$ *and* $q_2 \to \pi_2(P)$,
- $P \to (q_1, q_2)$ *if* $\pi_1(P) \to q_1$ *and* $\pi_2(P) \to q_2$.

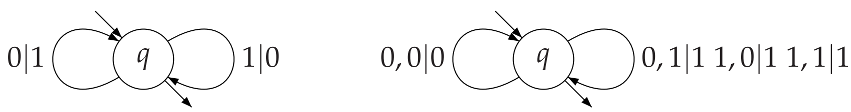
**DEFINITION 5.** *Suppose now that the output alphabet of $\mathcal{A}_1$ is the input alphabet of $\mathcal{A}_2$, i.e. $\Gamma = \Sigma'$. The* composition *of $\mathcal{A}_1$ and $\mathcal{A}_2$ is the automaton $\mathcal{A}_2 \circ \mathcal{A}_1 = (Q_1 \times Q_2, \Sigma, \Delta, \delta, I_1 \times I_2, F_1 \times F_2)$. The transitions in $\delta$ are:*

- $(q_1, q_2) \xrightarrow{a|c} (q_1', q_2')$ *if* $q_1 \xrightarrow{a|b} q_1'$ *and* $q_2 \xrightarrow{b|c} q_2'$,
- $(q_1, q_2) \to P$ *if* $q_1 \to \pi_1(P)$ *and* $q_2 \to \pi_2(P)$,
- $P \to (q_1, q_2)$ *if* $\pi_1(P) \to q_1$ *and* $\pi_2(P) \to q_2$.

Recall that LTL formulae are given by $\varphi := p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathcal{U} \varphi \mid \varphi \mathcal{U}' \varphi \mid \varphi \mathcal{S} \varphi \mid \varphi \mathcal{S}' \varphi$. For each atomic proposition $p$ we construct an automaton $\mathcal{A}_p$ which, given a word $x$, outputs $v_p(x)$. For each logical connective of arity $n$, we construct an automaton with input alphabet $\{0,1\}^n$, and output alphabet $\{0,1\}$. The input word is the tuple of truth words of the connective's variables, the output is the truth word of the complete formula. For temporal connectives, we only describe the automata corresponding to $\mathcal{U}$ and $\mathcal{U}'$. For the "past" connectives, the automata are the same with all transitions (successor and limits) reversed, and initial and final states swapped.

For any $p \in \mathrm{AP}$, the automaton $\mathcal{A}_p$ is $(\{q\}, 2^{\mathrm{AP}}, \{0,1\}, \delta, \{q\}, \{q\})$ where $\delta = \{(q \xrightarrow{a|0} q \mid p \notin a\} \cup \{q \xrightarrow{a|1} q \mid p \in a\} \cup \{q \to \{q\}, \{q\} \to q\}$. This automaton simply outputs 1 at positions where $p$ is true, and 0 everywhere else. Note that the run is uniquely determined by the input word; such a transducer is called non-ambiguous.

Figures 2(a) and 2(b) show the automata corresponding to the negation ($\neg$) and disjunction ($\vee$) connectives. Their limit transitions are $\{q\} \to q$ and $q \to \{q\}$. Again, these automata admit exactly one run for each input word.




(a) Automaton for negation (b) Automaton for disjunction

### 3.2 Automaton for $\mathcal{U}$

The difficulty starts with the "Until" connective ($\mathcal{U}$). We recall that $\varphi\mathcal{U}\psi$ holds at position $i$ in a word $w$ if there exists $j > i$ such that $\psi$ holds at $j$, and such that $\varphi$ holds at every position $k$ such that $i < k < j$.

We build an automaton $\mathcal{A}_\mathcal{U}$ with input alphabet $\{0,1\}^2$ (corresponding to the truth value of $\varphi$ and $\psi$ at each position), and output alphabet $\{0,1\}$. On an input word of the form $(v_\varphi(w), v_\psi(w))$ for some word $w$, we want the output to be $v_{\varphi\mathcal{U}\psi}(w)$. Let $J = |w|$, and $c \in \hat{J}$. We have five different situations. For each of these cases the figure shows an example, with "$|$" representing the cut $c$, and each $\bullet$ representing a position in the input word.

0. $c$ is followed by a position where $\varphi$ and $\psi$ are true. $\quad\begin{array}{ll} \text{input} & \cdots \bullet \mid \overset{1,1}{\bullet} \cdots \\ \text{output} & \quad\; 1 \end{array}$
1. $c = c_j^-$, and $j$ is such that $\varphi$ is false and $\psi$ is true. $\quad\begin{array}{ll} \text{input} & \cdots \bullet \mid \overset{0,1}{\bullet} \cdots \\ \text{output} & \quad\; 1 \end{array}$
2. other cases where $\varphi\mathcal{U}\psi$ is true at $c$. $\quad\begin{array}{ll} \text{input} & \cdots \bullet \mid \overbrace{\cdots}^{1,-} \overset{-,1}{\bullet} \cdots \\ \text{output} & \quad\; 1 \end{array}$
3. $c$ is followed by a position where both $\varphi$ and $\psi$ are false. $\quad\begin{array}{ll} \text{input} & \cdots \bullet \mid \overset{0,0}{\bullet} \cdots \\ \text{output} & \quad\; 0 \end{array}$
4. other cases where $\varphi\mathcal{U}\psi$ is false at $c$. If $c = c_j^-$ then the input at position $j$ is $(1,0)$.

$$\begin{array}{ll} \text{input} & \cdots \bullet \mid \overbrace{\cdots}^{1,0} \overset{0,0}{\bullet} \cdots \\ \text{output} & \quad\; 0 \end{array} \qquad \begin{array}{l} \cdots \bullet \mid \overbrace{\cdots}^{1,0} \; \overbrace{\cdots\cdots}^{\{(1,-),(0,1)\}} \; \cdots \\ \quad\; 0 \end{array}$$

The structure of the automaton $\mathcal{A}_\mathcal{U}$ and the limit transitions are given by Figure 2. This automaton has five states $q_0$ to $q_4$ corresponding to the situations described above. Given any two states $q$ and $q'$ there exists a transition $q \to q'$ except from $q_2$ to $q_3$ or $q_4$ and from $q_4$ to $q_0$, $q_1$ or $q_2$. The input label of successor transitions is determined by the origin node: $(1,1)$ for $q_0$, $(0,1)$ for $q_1$, $(0,0)$ for $q_3$, and $(1,0)$ for $q_2$ and $q_4$. The output label is 1 on transitions leading to $q_0$, $q_1$ or $q_2$, and 0 on transitions leading to $q_3$ or $q_4$. All states are initial, while $q_4$ is the only final state.

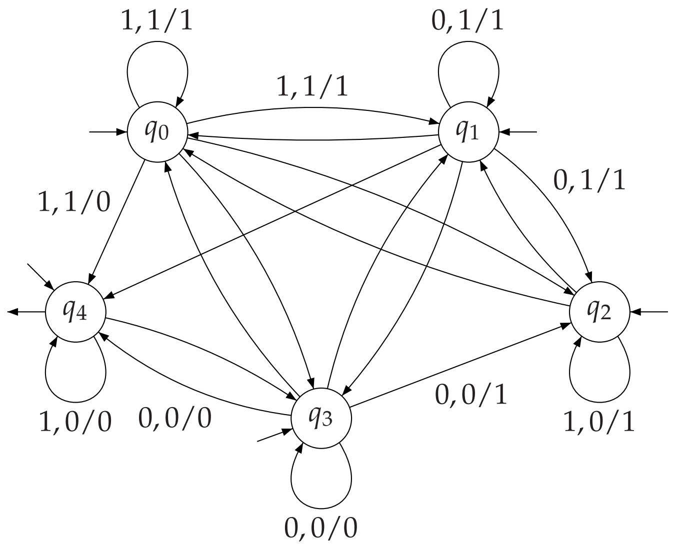
**LEMMA 6.** *Let $\varphi$ and $\psi$ two formulae. Let $x$ and $y$ be the truth words of $\varphi$ and $\psi$ on a word $w$ of length $J$. The output of $\mathcal{A}_\mathcal{U}$ on $(x, y)$ is the truth word of $\varphi\mathcal{U}\psi$ on $w$.*

PROOF. Let $\rho$ be the word of length $\hat{J}$ on $Q$ defined by

- if $x_j = y_j = 1$, then $\rho(c_j^-) = q_0$;
- if $x_j = 0$ and $y_j = 1$ then $\rho(c_j^-) = q_1$;
- if $x_j = y_j = 0$ then $\rho(c_j^-) = q_3$;
- otherwise, if there exists $j > c$ such that $y_j = 1$ and for all $i$ such that $c < i < j$, $x_i = 1$, then $\rho(c) = q_2$;
- otherwise, $\rho(c) = q_4$.

We show that $\rho$ is a run of $\mathcal{A}_\mathcal{U}$, that it is unique, and that its output is indeed the truth word of $\varphi\mathcal{U}\psi$ on $w$.

By definition, $\rho$ ends in $q_4$, which is the final state of $\mathcal{A}_\mathcal{U}$. Let $c \in \hat{J}$. If $\rho(c)$ is $q_0$, $q_1$ or $q_3$, then $c = c_j^-$ for some $j$ and the successor transition from $c$ to the next cut is allowed by the automaton. If $\rho(c) = q_2$, and $c = c_j^-$ for some $j$, then $x_j = 1$ and $y_j = 0$, and $\rho(c_j^+)$ is $q_0$,




$$
\begin{array}{rcl}
P & \to & q_0, q_1, q_2, q_3, q_4 \\
 & & \text{if } q_0,\, q_1 \text{ or } q_3 \in P \\
\{q_2\} & \to & q_0, q_1, q_2 \\
\{q_4\} & \to & q_3, q_4 \\
q_2 & \to & \{q_0\}, \{q_2\}, \{q_0, q_2\} \\
q_4 & \to & P \text{ if } q_1 \text{ or } q_3 \in P \\
q_4 & \to & \{q_4\}
\end{array}
$$

Figure 2: Automaton for $\mathcal{U}$

$q_1$ or $q_2$. If $\rho(c_j^-) = q_4$, then similarly $x_j = 1$ and $y_j = 0$, and $\rho(c_j^+)$ can be $q_3$ or $q_4$. Every successor transition in $\rho$ is thus allowed by $\mathcal{A}_{\mathcal{U}}$.

We now need to show the same for limit transitions. If a left limit transition leads to a cut $c$, then either $\psi$ is true arbitrarily close to the left of $c$ (in which case the corresponding limit set contains $q_0$ or $q_1$), or it is always false (and the limit set is $\{q_2\}$ or a subset of $\{q_3, q_4\}$). If the limit set contains $q_0$, $q_1$ or $q_3$, any state for $c$ is allowed. If it is $\{q_2\}$, the cut $c$ can't be labelled by $q_3$ or $q_4$ without violating the definition of $\rho$. Conversely, if the limit set is $\{q_4\}$, $\rho(c)$ is necessarily $q_3$ or $q_4$.

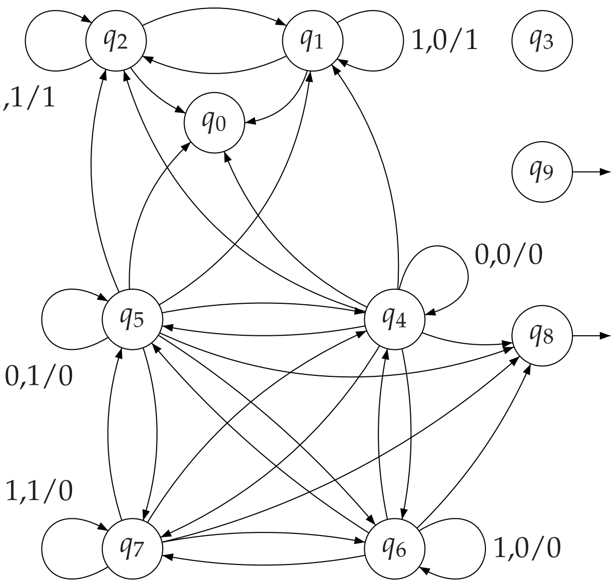
Let's now consider a right limit transition starting at a cut $c$. The label of this cut can only be $q_2$ or $q_4$. In the first case, $\varphi$ must be true everywhere in the limit set, which is thus a subset of $\{q_0, q_2\}$. In the second case, either $\varphi$ is false infinitely often in the limit, or $\psi$ is always false. This means that the limit set contains $q_1$ or $q_3$, or is restricted to $\{q_4\}$.

We now show that a run on $\mathcal{A}_{\mathcal{U}}$ is uniquely determined by the input word. Let $\gamma$ a run of $\mathcal{A}_{\mathcal{U}}$ on $x, y$. Because of the constraints on the successor transitions, a cut $c$ is labelled by $q_0$, $q_1$ or $q_3$ in $\gamma$ if and only if it is labelled by the same state in $\rho$.

Let's suppose that a cut $c$ is labelled by $q_2$ in $\gamma$. Since $q_2$ is not final, there exists $c' > c$ labelled by some other state. If there is a first such cut, its label is necessarily $q_0$ or $q_1$ (by a successor transition from $q_2$ or a limit transition from $\{q_2\}$). Otherwise, there is a transition of the form $q_2 \to \{q_0\}$ or $q_2 \to \{q_0, q_2\}$. In both cases, $c$ satisfies the condition for cuts labelled by $q_2$ in the definition of $\rho$. A similar argument shows that a cut labelled by $q_4$ in $\gamma$ has the same label in $\rho$. The run of $\mathcal{A}_{\mathcal{U}}$ on a given input word is thus unique.

The last step is to show that the output word is really the truth word of $\varphi\mathcal{U}\psi$. Let $j$ an element of $J$. First, suppose that $w, j \models \varphi\mathcal{U}\psi$. If $j$ has a successor $k$, and $\psi$ is true at $k$, then $y_k = 1$, and $\mathcal{A}_{\mathcal{U}}$ outputs 1 at position $j$. Otherwise, there exists $k > j$ such that $w, k \models \psi$ (i.e. $y_k = 1$), and $x_\ell = 1$ whenever $j < \ell < k$. Thus, $c_j^+$ is labelled with $q_2$, and $\mathcal{A}_{\mathcal{U}}$ once again outputs 1 at position $j$.

Similarly, if $w, j \not\models \varphi\mathcal{U}\psi$, there are two cases. In the first case, $j$ has a successor $k$, and $x_k = y_k = 0$. This means that $c_j^+$ is labelled by $q_3$, so the output at position $j$ is 0. In the last




- $P \to q_0, q_1, q_2$ if $P \cap \{q_4, q_5\} \neq \emptyset$ or $P \subseteq \{q_0, q_1, q_2\}$
- $P \to q_3$ if $P \subseteq \{q_0, q_1, q_2\}$
- $P \to q_4, q_5, q_6, q_7$ if $P \not\subseteq \{q_0, q_1, q_2\}$
- $P \to q_8$ if $P \cap \{q_4, q_5\} \neq \emptyset$
- $P \to q_9$ if $P \cap \{q_4, q_5\} = \emptyset$ and $P \not\subseteq \{q_0, q_1, q_2\}$
- $q_0 \to P$ if $P \subseteq \{q_0, q_1, q_2\}$
- $q_3 \to P$ if $P \cap \{q_1, q_4, q_6\} = \emptyset$ and $q_5 \in P$
- $q_8 \to P$ if $P \cap \{q_4, q_5, q_6, q_7\} \neq \emptyset$
- $q_9 \to P$ if $P \cap \{q_4, q_5, q_6, q_7\} \neq \emptyset$ and either $P \cap \{q_4, q_5\} = \emptyset$ or $P$ intersects $\{q_1, q_4, q_6\}$

Figure 3: Automaton for the future Stavi operator

case, $c_j^+$ is labelled by $q_4$, and once again $\mathcal{A}_{\mathcal{U}}$ outputs 0. ∎

### 3.3 Automaton for the future Stavi connective ($\mathcal{U}'$)

Let's recall that $\varphi\mathcal{U}'\psi$ holds at position $i$ if there exists a gap $c > i$ such that $\varphi$ holds at every position $i < j < c$, the property $\psi$ holds at every position in some interval starting at $x$, and $\neg\varphi$ holds at positions arbitrarily close to $c$ to the right.

The central point in this definition is the gap $c$, which corresponds to state $q_3$ in the automaton. States $q_0$, $q_1$ and $q_2$ follow the positions, before $q_3$, where the formula holds. States $q_4$, $q_5$, $q_6$, $q_7$, $q_8$ follow the positions where the formula doesn't hold. If a run reaches $q_0$, $q_1$ or $q_2$, it has to leave this region through $q_3$, and all successor transitions until then have input label $(1,0)$ or $(1,1)$. The structure of this automaton is depicted in Figure 3. All states except $q_3$ and $q_9$ are initial; $q_8$ and $q_9$ are final. Transitions from $q_1$ and $q_7$ have input label $(1,1)$, transitions from $q_2$ and $q_6$ have input label $(1,0)$, transitions from $q_4$ have input label $(0,0)$, and transitions from $q_5$ have input label $(0,1)$. The output is 1 for transitions to $q_0$, $q_1$ and $q_2$, and 0 for transitions to $q_4$, $q_5$, $q_6$, $q_7$ and $q_8$.

We define a labelling $\rho$ of the cuts of a word $w$ on $\{0,1\}^2$ using the states of the automaton in the following way:

- $q_0$ has no successor, $\varphi\mathcal{U}'\psi$ is true
- $q_1$ has an outgoing transition labelled $(1,0)$, $\varphi\mathcal{U}'\psi$ is true
- $q_2$ has an outgoing transition labelled $(1,1)$, $\varphi\mathcal{U}'\psi$ is true
- $q_3$ is a gap, $\varphi\mathcal{U}'\psi$ is true before it and false afterwards
- $q_4$ has an outgoing transition labelled $(0,0)$, $\varphi\mathcal{U}'\psi$ is false
- $q_5$ has an outgoing transition labelled $(0,1)$, $\varphi\mathcal{U}'\psi$ is false
- $q_6$ has an outgoing transition labelled $(1,0)$, $\varphi\mathcal{U}'\psi$ is false
- $q_7$ has an outgoing transition labelled $(1,1)$, $\varphi\mathcal{U}'\psi$ is false

- $q_8$ has no successor, $\varphi$ doesn't hold in the left limit if it has no predecessor, and $\varphi\mathcal{U}'\psi$ is false
- $q_9$ is a gap or is the last cut, $\varphi\mathcal{U}'\psi$ is false, and $\varphi$ is true in some interval to the left

**LEMMA 7.** *$\rho$ defines the unique run of the automaton on its input word. If the input is $(v_\varphi(w), v_\psi(w))$ for some word $w$, then the output of this run is $v_{\varphi\mathcal{U}'\psi}(w)$.*

PROOF. We first show that $\rho$ is a run. Successor transitions correspond almost directly to the definitions of the labelling $\rho$, so let's look at limit transitions. For left limits, the following cases need to be considered:

- if a transition $P \to q_0$ is taken at a cut $c$, then either $\varphi$ is true in the limit, and so $\varphi\mathcal{U}'\psi$ is too, and $P \subseteq \{q_0, q_1, q_2\}$, or it's not, and either $q_4$ or $q_5$ appear in the limit
- the same reasoning applies for $q_1$ and $q_2$
- if $c$ is labelled $q_3$ then the incoming transition has to come from a subset of $\{q_0, q_1, q_2\}$ since $\varphi\mathcal{U}'\psi$ is true in the limit.
- if a transition $P \to q_4$ is used, then $\varphi\mathcal{U}'\psi$ is not true in the limit (otherwise it would still be true), and so $P \not\subseteq \{q_0, q_1, q_2\}$; the same applies for $q_5$, $q_6$, $q_7$, $q_8$ and $q_9$
- if $c$ is a left limit and is labelled $q_8$ then the incoming transition comes from a set $P$ intersecting $\{q_4, q_5\}$ because $\neg\varphi$ is repeated
- if $c$ is labelled $q_9$ then $q_4$ and $q_5$ can't appear in the left limit set ($\varphi$ is true)

If $c$ is a right limit cut, it can only be labelled $q_0$, $q_3$, $q_8$ or $q_9$. Here are the possible right-limit transitions:

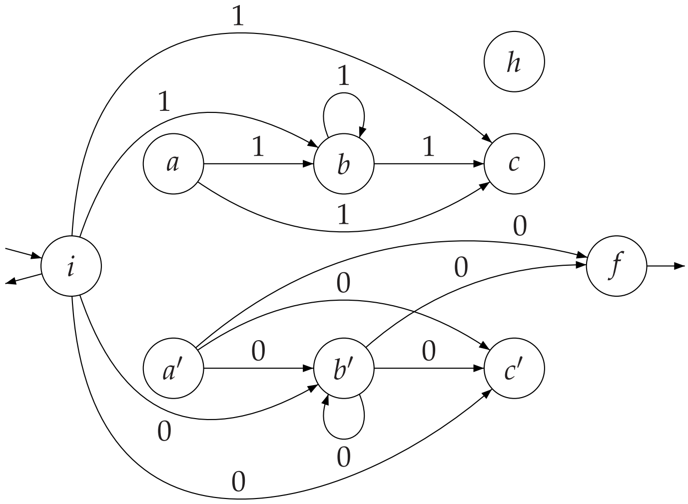
- if a right-limit cut $c$ is labelled $q_0$, the limit transition has to go to a subset of $\{q_0, q_1, q_2\}$ since $\varphi\mathcal{U}'\psi$ holds in the limit
- if $c$ is labelled with $q_3$, the limit transition to its right leads necessarily to a set $P$ not including $q_1$, $q_4$ and $q_6$ since $\psi$ is always true, and including $q_5$ because $\neg\varphi$ is repeated
- if $c$ is labelled $q_8$ or $q_9$, the right limit set can't be a subset of $\{q_0, q_1, q_2\}$ otherwise $c$ would have been labelled $q_0$
- if $c$ is labelled $q_9$ we have the additional condition that either $\varphi$ holds in the limit (and neither $q_4$ nor $q_5$ appears) or $\psi$ doesn't (and one of $q_1$, $q_4$ and $q_6$ is in the limit)

The labelling of cuts defined above is thus a path of the automaton, and we only need to show that it's the only one, using the same method as for the $\mathcal{A}_\mathcal{U}$. Moreover, the definition of $\rho$ means that the output is 1 whenever $\varphi\mathcal{U}'\psi$ holds, and 0 at all other positions. ∎

### 3.4 Construction of $\mathcal{A}_\varphi$

Now that we have the basic blocks for our construction, we can build an automaton for any formula $\varphi$. If $\varphi$ is an atomic proposition $p$, then we have seen how to build $\mathcal{A}_p$ in the previous section. If $\varphi = \neg\psi$, then $\mathcal{A}_\varphi = \mathcal{A}_\neg \circ \mathcal{A}_\psi$. If $\varphi = \psi_1 \vee \psi_2$, then $\mathcal{A}_\varphi = \mathcal{A}_\vee \circ (\mathcal{A}_{\psi_1} \times \mathcal{A}_{\psi_2})$. If $\varphi = \psi_1\mathcal{U}\psi_2$, then $\mathcal{A}_\varphi = \mathcal{A}_\mathcal{U} \circ (\mathcal{A}_{\psi_1} \times \mathcal{A}_{\psi_2})$. The same can be done for $\mathcal{U}'$ and for the past connectives.

The number of states of the resulting automaton is the product of the number of states of all the elementary automata, and is thus exponential in the size of the formula. The actual size of the automaton includes limit transitions, so can be doubly exponential in the size of the formula, if those transitions are represented explicitly.




Limit transitions:

$P \to a, h$ if $P \subseteq \{a, b, c, h\}$

$c \to P$ if $P \subseteq \{a, b, c, h\}$

$P \to a', f$ for any $P$

$c' \to P$ if $P \cap \{a, b, c, h\} = \emptyset$

Figure 4: Automaton checking whether a gap exists in the future

To check whether the formula $\varphi$ is satisfiable by a model which is recognized by an automaton $\mathcal{B}$, we can compute the product of the automaton $\mathcal{A}_\varphi$ with $\mathcal{B}$, and check whether a transition where $\mathcal{A}_\varphi$ outputs 1 is accessible and co-accessible. This ensures that there exists a successful run of the product automaton going through that transition, meaning that the corresponding input word is accepted by $\mathcal{B}$ and there is a position where $\varphi$ holds. This concludes the proof of Corollary 3.

## 4 Discussion

**Logical characterization of automata.** We have shown that any LTL, and thus FO, formula can be represented as a non-ambiguous automaton with output. But one can also build such an automaton where the output is the truth word of a property which can't be expressed as a first-order formula. The automaton shown on Figure 4 outputs 1 whenever "there is a gap somewhere in the future" is true; that formula can't be expressed in FO. It would be interesting to find a logical characterisation of the properties that can be expressed using such automata.

**Computational complexity.** The exact complexity of the satisfiability problem for LTL on arbitrary orderings remains open. We give a 2EXPSPACE procedure to compute an automaton from a formula, whose emptiness can then be checked efficiently. A classical optimization in similar problems is to compute the automaton on the fly, which saves a lot of complexity, so an algorithm using this technique for LTL on arbitrary orderings would be interesting.

**Expressive power.** On finite and $\omega$-words, LTL restricted to the unary operators ($\mathcal{X}$, $\mathcal{F}$, and their past counterparts) is equivalent to first-order logic restricted to two variables, $FO^2(<, +1)$ [8]. Restricting even further to $\mathcal{F}$ and its reverse, we get a logic expressively equivalent to $FO^2(<)$. In the case of finite words, $FO^2(<)$ corresponds to "partially ordered" two-way automata [16]. The proof of equivalence between unary temporal logic and $FO^2$ can be easily extended to the case of arbitrary linear orderings. It would be interesting to find such a correspondence for arbitrary orderings as well, and to see if these restrictions provide lower complexity results.

**Mosaics technique.** In his work on LTL($\mathcal{U}$), Reynolds uses "mosaics" to keep track of the subformulas that need to be satisfied in particular intervals, and to find a decomposition that shows the satisfiability of the initial formula. Unfortunately it is not clear if and how this can be extended to handle a larger fragment of the logic.

## 5 Conclusion

We investigate linear temporal order with Until, Since, and the Stavi connectives over general linear time, and its relationship with automata over linear orderings. We provide a translation from LTL to a class of non-ambiguous automata with output, giving a 2EXPSPACE procedure to decide satisfiability of a formula in any rational subclass.

This leaves a number of immediate questions, starting with the actual complexity for the satisfiability problem for LTL, but also for some of its fragments, where some operators are excluded. While the full class of automata over linear orderings is not closed under complementation [1], it might still be possible to find a logical characterization for some interesting subclasses.

## References


[1] Nicolas Bedon, Alexis Bès, Olivier Carton, and Chloe Rispal. Logic and rational languages of words indexed by linear orderings. In Edward A. Hirsch, Alexander A. Razborov, Alexei L. Semenov, and Anatol Slissenko, editors, *CSR*, volume 5010 of *Lecture Notes in Computer Science*, pages 76–85. Springer, 2008.

[2] Alexis Bès and Olivier Carton. A Kleene theorem for languages of words indexed by linear orderings. *Int. J. Found. Comput. Sci.*, 17(3):519–542, 2006.

[3] Véronique Bruyère and Olivier Carton. Automata on linear orderings. In Jiri Sgall, Ales Pultr, and Petr Kolman, editors, *MFCS*, volume 2136 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 2001.

[4] J. Richard Büchi. Transfinite automata recursions and weak second order theory of ordinals. pages 2–23, 1965.

[5] Olivier Carton. Accessibility in automata on scattered linear orderings. In Krzysztof Diks and Wojciech Rytter, editors, *MFCS*, volume 2420 of *Lecture Notes in Computer Science*, pages 155–164. Springer, 2002.

[6] Olivier Carton, 2009. Private communication.

[7] Stéphane Demri and Alexander Rabinovich. The complexity of temporal logic with until and since over ordinals. In Nachum Dershowitz and Andrei Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2007.

[8] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.

[9] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal basis of fairness. In *POPL*, pages 163–173, 1980.

[10] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *CAV*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.

[11] Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.

[12] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[13] Mark Reynolds. The complexity of the temporal logic with "until" over general linear time. *J. Comput. Syst. Sci.*, 66(2):393–426, 2003.

[14] Chloe Rispal and Olivier Carton. Complementation of rational sets on countable scattered linear orderings. *Int. J. Found. Comput. Sci.*, 16(4):767–786, 2005.

[15] J. G. Rosenstein. *Linear Orderings*. Academic Press, New York, 1982.

[16] Thomas Schwentick, Denis Thérien, and Heribert Vollmer. Partially-ordered two-way automata: A new characterization of DA. In *DLT '01: Revised Papers from the 5th International Conference on Developments in Language Theory*, pages 239–250, London, UK, 2002. Springer-Verlag.

[17] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.

[18] Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, MIT, Cambridge, Massasuchets, USA, 1974.

[19] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344. IEEE Computer Society, 1986.

[20] Pierre Wolper. The tableau method for temporal logic: an overview. In *Logique et Analyse*, pages 28–119, 1985.