On-the-fly Automata Construction for Dynamic Linear Time Temporal Logic

Laura Giordano
Università del Piemonte Orientale
Alessandria, Italy
laura@mfn.unipmn.it

Alberto Martelli
Università di Torino
Torino, Italy
mrt@di.unito.it

Abstract

We present a tableau-based algorithm for obtaining a Büchi automaton from a formula in Dynamic Linear Time Temporal Logic ($DLTL$), a logic which extends LTL by indexing the until operator with regular programs. The construction of the states of the automaton is similar to the standard construction for LTL, but a different technique must be used to verify the fulfillment of until formulas. The resulting automaton is a Büchi automaton rather than a generalized one. The construction can be done on-the-fly, while checking for the emptiness of the automaton.

1. Introduction

The problem of constructing automata from Linear-Time Temporal (LTL) formulas has been deeply studied [11]. The interest on this problem comes from the wide use temporal logic for the verification of properties of concurrent systems. The standard approach to LTL model checking consists of translating the negation of a given LTL formula (property) into a Büchi automaton, and checking the product of the property automaton and the model for language emptiness. Therefore it is essential to keep the size of the automaton as small as possible. A tableau-based algorithm for efficiently constructing a Büchi automaton is presented in [2]. This algorithm allows to build the graph "on the fly" and in most cases builds quite small automata, although the problem is intrinsically exponential. Further improvements have been presented in [1, 8].

Dynamic Linear Time Temporal Logic ($DLTL$) [6] extends LTL by indexing the until operator with programs in Propositional Dynamic Logic, and has been shown to be strictly more expressive than LTL [6]. In [3, 4] we have developed an action theory based on DLTL and of its product version [5], and we have shown how to use it to model multi-agent systems and to verify their properties, in particular by using model checking techniques. In [6] it is shown that the satisfiability problem for DLTL can be solved in exponential time, by reducing it to the emptiness problem for Büchi automata. This motivates the interest in developing efficient techniques for translating formulas into automata.

In this paper we present an efficient tableau-based algorithm for constructing a Büchi automaton from a DLTL formula. The construction of the states of the automaton is similar to the standard construction for LTL [2], but the possibility of indexing until formulas with regular programs puts stronger constraints on the fulfillment of until formulas than in LTL, requiring more complex acceptance conditions. Thus we extend the structure of graph nodes and the acceptance conditions by adapting a technique proposed in [6]. The resulting automaton will be a Büchi automaton instead of a generalized Büchi automaton as in [2].

2. Dynamic Linear Time Temporal Logic

In this section we shortly define the syntax and semantics of DLTL as introduced in [6]. In such a linear time temporal logic the next state modality is indexed by actions. Moreover, (and this is the extension to LTL) the until operator is indexed by programs in Propositional Dynamic Logic (PDL).

Let Σ be a finite non-empty alphabet. The members of Σ are actions. Let Σ^* and Σ^ω be the set of finite and infinite words on Σ, where $\omega = \{0, 1, 2, \ldots\}$. Let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. We denote by σ, σ' the words over Σ^ω and by τ, τ' the words over Σ^*. Moreover, we denote by $\leq$ the usual prefix ordering over Σ^* and, for $u \in \Sigma^\infty$, we denote by *prf(u)* the set of finite prefixes of u.

We define the set of programs (regular expressions) $Prg(\Sigma)$ generated by Σ as follows:

$$Prg(\Sigma) ::= a \mid \pi_1 + \pi_2 \mid \pi_1; \pi_2 \mid \pi^*$$

where $a \in \Sigma$ and π_1, π_2, π range over $Prg(\Sigma)$. A set of finite words is associated with each program by the mapping $[[]] : Prg(\Sigma) \rightarrow 2^{\Sigma^*}$, which is defined as follows:

- $[[a]] = \{a\}$;

- $[[\pi_1 + \pi_2]] = [[\pi_1]] \cup [[\pi_2]]$;
- $[[\pi_1; \pi_2]] = \{\tau_1\tau_2 \mid \tau_1 \in [[\pi_1]] \text{ and } \tau_2 \in [[\pi_2]]\}$;
- $[[\pi^*]] = \bigcup[[\pi^i]]$, where
  - $[[\pi^0]] = \{\varepsilon\}$
  - $[[\pi^{i+1}]] = \{\tau_1\tau_2 \mid \tau_1 \in [[\pi]] \text{ and } \tau_2 \in [[\pi^i]]\}$, for every $i \in \omega$.

Let $\mathcal{P} = \{p_1, p_2, \ldots\}$ be a countable set of atomic propositions. The set of formulas of DLTL(Σ) is defined as follows:

$$\text{DLTL}(\Sigma) ::= p \mid \neg\alpha \mid \alpha \vee \beta \mid \alpha\mathcal{U}^\pi\beta$$

where $p \in \mathcal{P}$ and α, β range over DLTL(Σ).

A model of DLTL(Σ) is a pair $M = (\sigma, V)$ where $\sigma \in \Sigma^\omega$ and $V : prf(\sigma) \to 2^{\mathcal{P}}$ is a valuation function. Given a model $M = (\sigma, V)$, a finite word $\tau \in prf(\sigma)$ and a formula α, the satisfiability of a formula α at τ in M, written $M, \tau \models \alpha$, is defined as follows:

- $M, \tau \models p$ iff $p \in V(\tau)$;
- $M, \tau \models \neg\alpha$ iff $M, \tau \not\models \alpha$;
- $M, \tau \models \alpha \vee \beta$ iff $M, \tau \models \alpha$ or $M, \tau \models \beta$;
- $M, \tau \models \alpha\mathcal{U}^\pi\beta$ iff there exists $\tau' \in [[\pi]]$ such that $\tau\tau' \in prf(\sigma)$ and $M, \tau\tau' \models \beta$. Moreover, for every τ'' such that $\varepsilon \leq \tau'' < \tau'$[1], $M, \tau\tau'' \models \alpha$.

A formula α is satisfiable iff there is a model $M = (\sigma, V)$ and a finite word $\tau \in prf(\sigma)$ such that $M, \tau \models \alpha$.

The formula $\alpha\mathcal{U}^\pi\beta$ is true at τ if "α until β" is true on a finite stretch of behavior which is in the linear time behavior of the program π.

The derived modalities $\langle\pi\rangle$ and $[\pi]$ can be defined as follows: $\langle\pi\rangle\alpha \equiv \top\mathcal{U}^\pi\alpha$ and $[\pi]\alpha \equiv \neg\langle\pi\rangle\neg\alpha$.

Furthermore, if we let $\Sigma = \{a_1, \ldots, a_n\}$, the $\mathcal{U}$, O (next), $\Diamond$ and $\Box$ of LTL can be defined as follows: $O\alpha \equiv \bigvee_{a\in\Sigma}\langle a\rangle\alpha$, $\alpha\mathcal{U}\beta \equiv \alpha\mathcal{U}^{\Sigma^*}\beta$, $\Diamond\alpha \equiv \top\mathcal{U}\alpha$, $\Box\alpha \equiv \neg\Diamond\neg\alpha$, where, in $\mathcal{U}^{\Sigma^*}$, Σ is taken to be a shorthand for the program $a_1 + \ldots + a_n$. Hence both LTL(Σ) and PDL are fragments of DLTL(Σ). As shown in [6], DLTL(Σ) is strictly more expressive than LTL(Σ). In fact, as the logic ETL [10] to which DLTL is inspired, DLTL has the full expressive power of the monadic second order theory of ω-sequences.

3. Automaton Construction

In this section we show how to build a Büchi automaton for a given DLTL formula ϕ using a tableau-like procedure. The automaton generates all the infinite sequences (models) satisfying the formula ϕ. First we recall the definition of Büchi automata.

1 We define $\tau \leq \tau'$ iff $\exists\tau''$ such that $\tau\tau'' = \tau'$. Moreover, $\tau < \tau'$ iff $\tau \leq \tau'$ and $\tau \neq \tau'$.

A *Büchi automaton* over an alphabet Σ is a tuple $\mathcal{B} = (Q, \to, Q_{in}, F)$ where:

- Q is a finite nonempty set of states;
- $\to \subseteq Q \times \Sigma \times Q$ is a transition relation;
- $Q_{in} \subseteq Q$ is the set of initial states;
- $F \subseteq Q$ is a set of accepting states.

Let $\sigma \in \Sigma^\omega$. Then a *run* of $\mathcal{B}$ over σ is a map $\rho : prf(\sigma) \to Q$ such that:

- $\rho(\varepsilon) \in Q_{in}$
- $\rho(\tau) \xrightarrow{a} \rho(\tau a)$ for each $\tau a \in prf(\sigma)$

The run ρ is *accepting* iff $inf(\rho) \cap F \neq \emptyset$, where $inf(\rho) \subseteq Q$ is given by: $q \in inf(\rho)$ iff $\rho(\tau) = q$ for infinitely many $\tau \in prf(\sigma)$. Finally $\mathcal{L}(\mathcal{B})$, the language of ω-words accepted by $\mathcal{B}$, is: $\mathcal{L}(\mathcal{B}) = \{\sigma | \exists$ an accepting run of $\mathcal{B}$ over $\sigma\}$.

Our aim is now to construct a Büchi automaton for a given DLTL formula ϕ. We build a graph defining the states and transitions of the automaton. A tableau-like procedure allows a node to be expanded by applying propositional rules as well as by expanding the temporal operators. It will make use of a reformulation of the following axioms of DLTL in [6][2]:

$\bigvee_{a\in\Sigma}\langle a\rangle\top$

$\alpha\mathcal{U}^\pi\beta \equiv (\beta \vee (\alpha \wedge \bigvee_{a\in\Sigma}\langle a\rangle \bigvee_{\pi'\in\delta_a(\pi)} \alpha\mathcal{U}^{\pi'}\beta))$, for $\varepsilon \in [[\pi]]$,

$\alpha\mathcal{U}^\pi\beta \equiv \alpha \wedge \bigvee_{a\in\Sigma}\langle a\rangle \bigvee_{\pi'\in\delta_a(\pi)} \alpha\mathcal{U}^{\pi'}\beta$, for $\varepsilon \notin [[\pi]]$,

where

$\delta_a(\pi) = \{\pi' | \pi \xrightarrow{a} \pi'\}$ and $\xrightarrow{a}$ is a transition relation (defined in [6]) such that the program π' is obtained from the program π by executing action a.

In our construction, we exploit the equivalence results between regular expressions and finite automata and we make use of an equivalent formulation of DLTL formulas in which "until" formulas are indexed with finite automata rather than regular expressions. Thus we have $\alpha\mathcal{U}^{\mathcal{A}}\beta$ instead of $\alpha\mathcal{U}^\pi\beta$, where $\mathcal{L}(\mathcal{A}) = [[\pi]]$. In fact, for each regular expression π there is an (ϵ-free) nondeterministic finite automaton $\mathcal{A}$, accepting the language $[[\pi]]$ generated by π. Moreover the size of the automaton is linear in the size of π [7]. Satisfiability of until formulas $\alpha\mathcal{U}^{\mathcal{A}}\beta$ must be modified accordingly by replacing $[[\pi]]$ with $\mathcal{L}(\mathcal{A})$ in the definition above[3].

More precisely, in the construction we will make use of the following notation for automata. Let $\mathcal{A} = (Q, \delta, Q_F)$ be an ϵ-free nondeterministic finite automaton over the alphabet Σ without an initial state, where Q is a finite set of

2 Remember that $\langle a\rangle\alpha \equiv \top\mathcal{U}^a\alpha$.

3 The idea of using finite state automata to label "until" formulas is inspired both to the automata construction for DLTL in [6] and to the automata construction for ETL in [9].

states, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, and Q_F is the set of final states. Given a state $q \in Q$, we denote with $\mathcal{A}(q)$ an automaton $\mathcal{A}$ with initial state q.

The two axioms above will thus be restated as follows:

$\alpha \mathcal{U}^{\mathcal{A}(q)} \beta \equiv (\beta \vee (\alpha \wedge \bigvee_{a \in \Sigma} \langle a \rangle \bigvee_{q' \in \delta(q,a)} \alpha \mathcal{U}^{\mathcal{A}(q')} \beta))$
(q is a final state)
$\alpha \mathcal{U}^{\mathcal{A}(q)} \beta \equiv \alpha \wedge \bigvee_{a \in \Sigma} \langle a \rangle \bigvee_{q' \in \delta(q,a)} \alpha \mathcal{U}^{\mathcal{A}(q')} \beta$
(q is not a final state)

These formulas can be easily proved to be valid. Observe that the disjunction $\bigvee_{q' \in \delta(q,a)} \alpha \mathcal{U}^{\mathcal{A}(q')} \beta$ is a finite disjunction, as the set of states q' in $\delta(q, a)$ is finite.

The main procedure to construct the Büchi automaton for a formula ϕ builds a graph $\mathcal{G}(\phi)$ whose nodes are labelled by sets of formulas, and which defines the states and the transitions of the Büchi automaton. The procedure makes use of an auxiliary tableau-based function which is described in the next section.

3.1. Tableau computation

The tableau procedure we introduce makes use of *signed formulas*, i.e. formulas prefixed with the symbol **T** or **F**. This procedure takes as input a set of formulas[4] and returns a set of sets of formulas, obtained by expanding the input set according to a set of tableau rules, formulated as follows:

$\phi \Rightarrow \psi_1, \psi_2$, if ϕ belongs to the set of formulas, then add ψ_1 and ψ_2 to the set

$\phi \Rightarrow \psi_1 | \psi_2$, if ϕ belongs to the set of formulas, then make two copies of the set and add ψ_1 to one of them and ψ_2 to the other one.

The rules are the following:

$\mathbf{T}(\alpha \wedge \beta) \Rightarrow \mathbf{T}\alpha, \mathbf{T}\beta$
$\mathbf{F}(\alpha \vee \beta) \Rightarrow \mathbf{F}\alpha, \mathbf{F}\beta$
$\mathbf{F}(\alpha \wedge \beta) \Rightarrow \mathbf{F}\alpha | \mathbf{F}\beta$
$\mathbf{T}(\alpha \vee \beta) \Rightarrow \mathbf{T}\alpha | \mathbf{T}\beta$
$\mathbf{T}\neg\alpha \Rightarrow \mathbf{F}\alpha$
$\mathbf{F}\neg\alpha \Rightarrow \mathbf{T}\alpha$
R1 $\mathbf{T}\alpha \mathcal{U}^{\mathcal{A}(q)} \beta \Rightarrow \mathbf{T}(\beta \vee (\alpha \wedge \bigvee_{a \in \Sigma} \langle a \rangle \bigvee_{q' \in \delta(q,a)} \alpha \mathcal{U}^{\mathcal{A}(q')} \beta))$
(q is a final state)
R2 $\mathbf{T}\alpha \mathcal{U}^{\mathcal{A}(q)} \beta \Rightarrow \mathbf{T}(\alpha \wedge \bigvee_{a \in \Sigma} \langle a \rangle \bigvee_{q' \in \delta(q,a)} \alpha \mathcal{U}^{\mathcal{A}(q')} \beta)$
(q is not a final state)
$\mathbf{F}\alpha \mathcal{U}^{\mathcal{A}(q)} \beta \Rightarrow \mathbf{F}(\beta \vee (\alpha \wedge \bigvee_{a \in \Sigma} \langle a \rangle \bigvee_{q' \in \delta(q,a)} \alpha \mathcal{U}^{\mathcal{A}(q')} \beta))$
(q is a final state)
$\mathbf{F}\alpha \mathcal{U}^{\mathcal{A}(q)} \beta \Rightarrow \mathbf{F}(\alpha \wedge \bigvee_{a \in \Sigma} \langle a \rangle \bigvee_{q' \in \delta(q,a)} \alpha \mathcal{U}^{\mathcal{A}(q')} \beta)$
(q is not a final state)

Given a set of formulas s, function *tableau(s)* works as follows:

- add $\mathbf{T} \bigvee_{a \in \Sigma} \langle a \rangle \top$ to s,
- expand the set of formulas of s according to the above rules (by possibly creating new sets) until all formulas in all sets have been expanded,
- return the resulting set of sets.

Formula $\mathbf{T} \bigvee_{a \in \Sigma} \langle a \rangle \top$ makes explicit that in DLTL each state must be followed by a next state ($O\top$ is an axiom in DLTL).

If the expansion of a set of formulas produces an inconsistent set, then this set is deleted (*consistency constraint*). A set is inconsistent if it contains either "$\mathbf{T}\bot$" or "$\mathbf{F}\top$" or "$\mathbf{T}\alpha$ and $\mathbf{F}\alpha$" or "$\mathbf{T}\langle a \rangle \alpha$ and $\mathbf{T}\langle b \rangle \beta$ with $a \neq b$".

Observe that the expansion of an until formula $\alpha \mathcal{U}^{\mathcal{A}(q)} \beta$ only requires a finite number of steps, namely a number of steps linear in the size of the automaton.

It is easy to see that for each set of formulas returned by *tableau* there is exactly one symbol $a \in \Sigma$ such that the set contains formulas of the form $\mathbf{T}\langle a \rangle \alpha$. In fact, because of $\mathbf{T} \bigvee_{a \in \Sigma} \langle a \rangle \top$, there must be at least one formula of that kind, whereas the consistency constraint prevents from having more than one formula of the form $\mathbf{T}\langle a \rangle \alpha$ for different symbols $a \in \Sigma$.

3.2. Building the graph

To build the graph we will consider each set of formulas obtained through the tableau construction as a node of the graph. The above tableau rules do not expand formulas of the kind $\langle a \rangle \alpha$. Since the operator $\langle a \rangle$ is a *next state* operator, expanding this kind of formulas from a node n means to create a new node containing α connected to n through an edge labelled with a. Given a node n containing a formula $\mathbf{T}\langle a \rangle \alpha$, then the set of nodes connected to n through an edge labelled a is given by $tableau(\{\mathbf{T}\alpha | \mathbf{T}\langle a \rangle \alpha \in n\} \cup \{\mathbf{F}\alpha | \mathbf{F}\langle a \rangle \alpha \in n\})$.

States and transitions of the Büchi automaton are obtained directly from the nodes and edges of the graph. While we will give later the details of the construction of the automaton, we want now to address the problem of defining accepting conditions. Intuitively this has to do with *until formulas*, i.e. formulas of the form $\mathbf{T}\alpha \mathcal{U}^{\mathcal{A}(q)} \beta$. If a node n of the graph contains the formula $\mathbf{T}\alpha \mathcal{U}^{\mathcal{A}(q)} \beta$, then we will accept an infinite path containing this node if it is followed in the path by a node n' containing $\mathbf{T}\beta$ and $\mathbf{T}\alpha \mathcal{U}^{\mathcal{A}(q_F)} \beta$, where q_F is a final state of $\mathcal{A}$. Furthermore if τ is the sequence of labels in the path from n to n', then τ must belong to $\mathcal{L}(\mathcal{A}(q))$, and all nodes between n and n' must contain $\mathbf{T}\alpha$.

This problem has been solved in LTL by imposing generalized Büchi acceptance conditions. In our formulation they could be stated as follows: For each subformula $\alpha \mathcal{U} \beta$ of the

4 In this section we will always refer to signed formulas

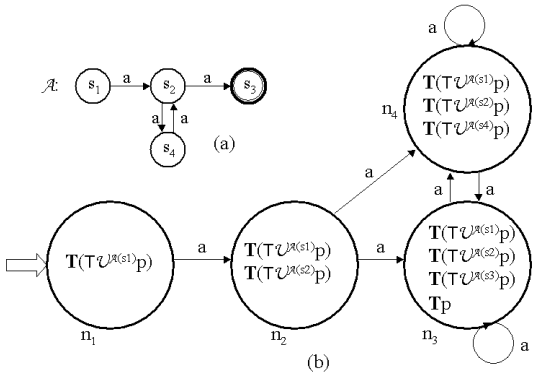

Figure 1. (a) automaton $\mathcal{A}$ and (b) graph for $\Box\langle\mathcal{A}(s_1)\rangle p$

initial formula there is a set F of accepting states including all the nodes $q \in Q$ such that either $\mathbf{T}\alpha\mathcal{U}\beta$ is not contained in the node or $\mathbf{T}\beta$ holds. Unfortunately a similar solution does not apply in the case of DLTL, because acceptance of until formulas is more constrained than in LTL.

Let us consider for instance the formula $\Box\langle\mathcal{A}(s_1)\rangle p$, with $\Sigma = \{a\}$. The automaton $\mathcal{A}$ is given in Figure 1(a). By eliminating the derived modalities, this formula can be rewritten as the signed formula $\mathbf{F}(\top\mathcal{U}^{\mathcal{A}_1(s_0)}\neg(\top\mathcal{U}^{\mathcal{A}(s_1)}p))$, where the automaton $\mathcal{A}_1$ has only one (final) state s_0 connected to itself through a transition labelled a. By applying the above construction starting from this formula, we obtain the graph in Figure 1(b), where for simplicity we have kept only the most significant formulas. Every node of this graph contains a formula $\mathbf{T}(\top\mathcal{U}^{\mathcal{A}(s_1)}p)$, and the only node which might fulfill the until formulas is node n_3, since it contains $\mathbf{T}(\top\mathcal{U}^{\mathcal{A}(s_3)}p)$, with s_3 final, and $\mathbf{T}p$. However it is easy to see that not all infinite paths through n_3 will be accepting. For instance, in the path $n_1, n_2, n_3, n_4, n_3, n_4, n_3, n_4, \ldots$ no occurrence of n_3 fulfills the formula $\mathbf{T}(\top\mathcal{U}^{\mathcal{A}(s_1)}p)$ in n_2, since the distance in this path between node n_2 and any occurrence of n_3 is odd, while all strings in $\mathcal{L}(\mathcal{A}(s_1))$ have even length.

We present now a different solution, derived from [6], where some of the nodes will be duplicated to avoid the above problem. Before describing the construction of the graph, we make the following observation. Let us assume that a node n contains the until formula $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$, such that q is not a final state. Since this formula has been expanded with (R2), node n will also contain $\mathbf{T}\langle a\rangle\bigvee_{q'\in\delta(q,a)}\alpha\mathcal{U}^{\mathcal{A}(q')}\beta$ for some a. Therefore, according to the construction of the successor nodes, each successor node will contain a formula $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q')}\beta$, where $q' \in \delta(q,a)$. We say that this until formula is *derived* from formula $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ in node n. On the other hand, if q is a final state, then $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ has been expanded with (R1), and two cases are possible: either n contains $\mathbf{T}\beta$ or all successor nodes contain a derived until formula as described above.

If a node contains an until formula which is not derived from a predecessor node, we will say that the formula is *new*. New until formulas are obtained during the expansion of the *tableau* procedure. It is easy to see that if $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ is a new formula, then $\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ must be a subformula of the initial formula. For instance, the formula $\mathbf{T}(\top\mathcal{U}^{\mathcal{A}(s_1)}p)$ is new in each of the nodes in Figure 1. Note that an until formula in a node might be both a derived and a new formula. In that case we will consider it as a derived formula.

We can now show how the graph can be built and how the accepting conditions are formulated. Each node of the graph is a triple $(\mathcal{F}, x, f)$, where $\mathcal{F}$ is an expanded set of formulas, $x \in \{0,1\}$, and $f \in \{\downarrow, \surd\}$.

In order to formulate the accepting condition, we must be able to trace the until formulas along the paths of the graph to make sure that they satisfy the until condition. Therefore we extend signed formulas so that all until formulas have a label 0 or 1, i.e. they have the form $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ where $l \in \{0,1\}$.

For each node $(\mathcal{F}, x, f)$, the label of an until formula in $\mathcal{F}$ will be assigned as follows. If it is a derived until formula, then its label is the same as that of the until formula in the predecessor node it derives from. Otherwise, if the formula is new, it is given the label $1 - x$. Of course function $tableau$ must be suitably modified in order to propagate the label from an until formula to its derived formulas in the successor nodes, and to give the right label to new formulas. To do this we assume that it has two parameters: a set of formulas and the value of x.

Function *create_graph* in Figure 2 builds a graph $\mathcal{G}(\phi)$, given an initial formula ϕ, by returning the triple $\langle Q, I, \Delta\rangle$, where Q is the set of nodes, I the set of initial nodes and $\Delta : Q \times \Sigma \times Q$ the set of labelled edges.

Note that two formulas $\mathbf{T}^0\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ and $\mathbf{T}^1\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ are considered to be different. For instance, by applying *create_graph* to the formula of Figure 1, we get two nodes

$(\{\mathbf{T}^0(\top\mathcal{U}^{\mathcal{A}(s_1)}p), \mathbf{T}^0(\top\mathcal{U}^{\mathcal{A}(s_2)}p), \mathbf{T}^1(\top\mathcal{U}^{\mathcal{A}(s_4)}p)\}, \downarrow, 1)$

and

$(\{\mathbf{T}^0(\top\mathcal{U}^{\mathcal{A}(s_1)}p), \mathbf{T}^0(\top\mathcal{U}^{\mathcal{A}(s_2)}p), \mathbf{T}^0(\top\mathcal{U}^{\mathcal{A}(s_4)}p),$
$\mathbf{T}^1(\top\mathcal{U}^{\mathcal{A}(s_2)}p)\}, \downarrow, 1)$.

These two nodes correspond to node n_4 in Figure 1.

States and transitions of the Büchi automaton $\mathcal{B}(\phi)$ are obtained directly from the nodes and edges of the graph. The set of accepting states consists of all states whose associated node contains $f = \surd$.

Let ρ be a run of $\mathcal{B}(\phi)$. Since we identify states of the automaton with nodes of the graph, ρ can also be considered as an infinite path of $\mathcal{G}(\phi)$, and $\rho(\tau)$ will denote a node of such a graph. According to the construction of the graph, the

function $create\_graph(\phi)$
$I := \emptyset$
for all $\mathcal{F} \in tableau(\{\mathbf{T}\phi\}, 0)$
  $I := I \cup \{(\mathcal{F}, 0, \surd)\}$
end-for
$U := Q := I$
$\Delta := \emptyset$
while $U \neq \emptyset$ **do**
  remove $n = (\mathcal{F}, x, f)$ from U
  if $f = \surd$ **then**
    $x' := 1 - x$
  else
    $x' := x$
  end-if
  for all $\mathcal{F}' \in tableau(\{\mathbf{T}\alpha | \mathbf{T}\langle a\rangle\alpha \in \mathcal{F}\} \cup \{\mathbf{F}\alpha | \mathbf{F}\langle a\rangle\alpha \in \mathcal{F}\}, x')$
    if $f = \surd$ **then**
      $f' :=\downarrow$
    else if there exists no $\mathbf{T}^{x'}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta \in \mathcal{F}'$ **then**
      $f' := \surd$
    else
      $f' :=\downarrow$
    end-if
    end-if
    $n' := (\mathcal{F}', x', f')$
    if $\exists n'' \in Q$ such that $n'' = n'$ **then**
      $\Delta := \Delta \cup \{(n, a, n'')\}$
    else
      $Q := Q \cup \{n'\}$
      $\Delta := \Delta \cup \{(n, a, n')\}$
      $U := U \cup \{n'\}$
    end-if
  end-for
end-while
return $\langle Q, I, \Delta\rangle$

Figure 2. Function *create_graph*

x and f values of the nodes of ρ have the following properties:

- if a node contains $(0, \surd)$ then its successor node contains $(1, \downarrow)$
- if a node contains $(1, \surd)$ then its successor node contains $(0, \downarrow)$
- if a node contains $(0, \downarrow)$ then its successor node contains either $(0, \downarrow)$ or $(0, \surd)$
- if a node contains $(1, \downarrow)$ then its successor node contains either $(1, \downarrow)$ or $(1, \surd)$

Therefore the sequence of the x and f values in ρ will be as follows:

$(0, \surd), (1, \downarrow), \ldots, (1, \downarrow), (1, \surd), (0, \downarrow), \ldots, (0, \downarrow), (0, \surd), \cdots$

Let us call *0-sequences* or *1-sequences* the sequences of nodes of ρ with $x = 0$ or $x = 1$ respectively. If ρ is an *accepting run*, then it must contain infinitely many nodes containing $\surd$, and thus all 0-sequences and 1-sequences must be finite.

Intuitively, every until formula contained in a node of a 0-sequence must be fulfilled within the end of the next 1-sequence, and vice versa. In fact, assuming that the formula has label 1, the label will be propagated to all derived formulas in the following nodes until a node is found fulfilling the until formula. But, on the other hand, the 1-sequence terminates only when there are no more until formulas with label 1, and thus that node must be met before the end of the next 1-sequence.

3.3. Correctness of the procedure

The next proposition summarizes what we have already pointed out in the previous section.

Proposition 1 *Assume that a node n of the graph contains $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$, and let a be the label of the outgoing edges (remember that all outgoing edges from a node have the same label). Then the following holds:*
if q is not a final state of $\mathcal{A}$

> *node n contains $\mathbf{T}\alpha$ and each outgoing edge leads to a node containing an until formula $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q')}\beta$ derived from $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ in n, such that $q' \in \delta(q, a)$*

else, if q is a final state of $\mathcal{A}$, either

> *(a) node n contains $\mathbf{T}\beta$ and no successor node contains a formula derived from $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$, or*
> *(b) node n contains $\mathbf{T}\alpha$ and each outgoing edge leads to a node containing a derived until formula $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q')}\beta$, such that $q' \in \delta(q, a)$*

Given a run ρ, we will denote with $\rho(\tau).\mathcal{F}$ the $\mathcal{F}$ field of the node $\rho(\tau)$, and similarly for the x and f fields.

Proposition 2 *Let $\sigma \in \Sigma^\omega$ and $\rho : prf(\sigma) \longrightarrow Q$ be a (non necessarily accepting) run. For each $\tau \in prf(\sigma)$, let $\rho(\tau) = (\mathcal{F}, x, f)$. Then for each $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q)}\beta \in \mathcal{F}$ one of the following holds:*

1. *$\forall\tau'$ s.t. $\tau\tau' \in prf(\sigma)$: $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q')}\beta \in \rho(\tau\tau').\mathcal{F}$ and $q' \in \delta^*_{\mathcal{A}}(q, \tau')$* [5]
2. *$\exists\tau'$ s.t. $\tau\tau' \in prf(\sigma)$: $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q')}\beta \in \rho(\tau\tau').\mathcal{F}$, q' is a final state, $q' \in \delta^*_{\mathcal{A}}(q, \tau')$, $\mathbf{T}\beta \in \rho(\tau\tau').\mathcal{F}$ and no*

5 $\delta^*_{\mathcal{A}}$ is the obvious extension of $\delta_{\mathcal{A}}$ to sequences

successor node of $\rho(\tau\tau')$ contains an until formula derived from $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q')}\beta$. Moreover, for every τ'' such that $\varepsilon \leq \tau'' < \tau', \mathbf{T}\alpha \in \rho(\tau\tau'').\mathcal{F}$.

For each $\mathbf{F}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta \in \mathcal{F}$ the following holds:

3. *$\forall\tau'$ s.t. $\tau\tau' \in prf(\sigma)$: if $\tau' \in \mathcal{L}(\mathcal{A}(q))$ then either $\mathbf{F}\beta \in \rho(\tau\tau').\mathcal{F}$ or there is τ'' such that $\varepsilon \leq \tau'' < \tau', \mathbf{F}\alpha \in \rho(\tau\tau'').\mathcal{F}$.*

Proof It follows from Proposition 1 and procedure *create_graph*.

In an accepting run, case (2) must hold for all until formulas and all nodes. This is proved in the following theorem, together with its converse.

Theorem 1 *Let $\sigma \in \Sigma^\omega$ and $\rho : prf(\sigma) \longrightarrow Q$ be a run. Then, for each $\tau \in prf(\sigma)$ and for each $\mathbf{T}^l\alpha\mathcal{U}^{\mathcal{A}(q)}\beta \in \rho(\tau).\mathcal{F}$, condition (2) of Proposition 2 holds if and only if ρ is an accepting run.*

Proof If part: ρ is an accepting run. As pointed out before the nodes of ρ are arranged in alternating 0-sequences and 1-sequences of finite length. Then we can have the following cases:

a) $l = 0$ and $\rho(\tau).x = 0$. Let us assume that condition (1) of Proposition 2 holds. Then each node $\rho(\tau\tau')$ following $\rho(\tau)$ in the same 0-sequence, will contain a derived formula $\mathbf{T}^0\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ (remember that the label of a derived formula cannot change). On the other hand, the 0-sequence containing $\rho(\tau)$ is finite, and, by construction, the last node of this sequence does not contain any until formula labelled with 0. Therefore the assumption is wrong, and condition (2) must hold.

b) $l = 1$ and $\rho(\tau).x = 1$. As case (a).

c) $l = 1$ and $\rho(\tau).x = 0$. Let us assume again that condition (1) of Proposition 2 holds. Then each node $\rho(\tau\tau')$ following $\rho(\tau)$ will contain an until formula derived from $\mathbf{T}^1\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ in $\rho(\tau)$. All derived formulas will be labelled 1 up to the last node of the 0-sequence. This label will necessarily propagate to the first node of the following 1-sequence, and we fall in case (b).

d) $l = 0$ and $\rho(\tau).x = 1$. As case (c).

Only if: condition (2) holds. We show that all 0 and 1-sequences of ρ are finite. This is true for the initial 0-sequence, which consists only of the first node. Let us assume now that a 0-sequence is finite. We show that the following 1-sequence is also finite. According to the construction, the last node of the 0-sequence can contain only until formulas with label 1. The following 1-sequence goes on until its nodes contain some until formula with label 1. Since condition (2) holds, for each of these until formulas there is a τ' such that the successor node of $\rho(\tau\tau')$ does not contain an until formula derived from it. On the other hand all new until formulas created in this 1-sequence will have label 0. Therefore, if τ^{max} is the longest among all τ', after node $\rho(\tau\tau^{max})$ there will be no until formula labelled with 1, and the 1-sequence will terminate. The same holds by replacing 0 with 1 and vice versa.

Lemma 1 *Let s be a set of formulas and $tableau(s) = \{s_1, \ldots, s_n\}$. Then $\bigwedge s \leftrightarrow \bigvee_{1\leq i\leq n} \bigwedge s_i$.*

Proof All rules used by the function $tableau$ correspond to equivalence formulas.

Lemma 2 *Let $M = (\sigma, V)$ be a model, $\tau \in prf(\sigma)$, and let $n = (\mathcal{F}, x, f)$ be a node of the graph such that $M, \tau \models \bigwedge \mathcal{F}$. Then there exists a successor $n' = (\mathcal{F}', x', f')$ of n such that $M, \tau a \models \bigwedge \mathcal{F}'$, where $\tau a \in prf(\sigma)$. Moreover, if $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta \in \mathcal{F}'$ where q is a final state and $M, \tau a \models \mathbf{T}\beta$, then $\mathbf{T}\beta \in \mathcal{F}'$.*

Proof The proof comes from the construction and the previous lemma. In particular the last part holds if, when expanding $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ in $\mathcal{F}'$ with rule (R1), we choose the set containing $\mathbf{T}\beta$.

Theorem 2 *Let $M = (\sigma, V)$ and $M, \varepsilon \models \phi$. Then $\sigma \in \mathcal{L}(\mathcal{B}(\phi))$.*

Proof We show how to build an accepting run ρ of $\mathcal{B}(\phi)$ over σ. The first node of ρ is chosen by taking an initial node $n = (\mathcal{F}, x, f)$ of the graph such that $M, \varepsilon \models \bigwedge \mathcal{F}$. The following nodes of ρ are chosen by repeatedly applying Lemma 2. To prove that the run is an accepting run, we have to show that all the until formulas are fulfilled. Assume that $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q)}\beta$ occurs on the run at $\rho(\tau)$. Then, for the choice of the run ρ, it must be that $M, \tau \models \alpha\mathcal{U}^{\mathcal{A}(q)}\beta$. By definition of satisfiability we have that there exists $\tau' \in \mathcal{L}(\mathcal{A}(q))$ such that $\tau\tau' \in prf(\sigma)$ and $M, \tau\tau' \models \beta$. Moreover, for every τ'' such that $\varepsilon \leq \tau'' < \tau'$, $M, \tau\tau'' \models \alpha$. As $\tau' \in \mathcal{L}(\mathcal{A}(q))$, by the choice of run ρ and the construction of the automaton, there must be a final state $q' \in \delta^*_{\mathcal{A}}(q, \tau')$ such that $\mathbf{T}\alpha\mathcal{U}^{\mathcal{A}(q')}\beta$ belongs to $\rho(\tau\tau').\mathcal{F}$. Moreover for all τ'' such that $\varepsilon \leq \tau'' < \tau'$, $\mathbf{T}\alpha$ belongs to $\rho(\tau\tau'').\mathcal{F}$. By Lemma 2, $\mathbf{T}\beta$ also belongs to $\rho(\tau\tau').\mathcal{F}$. Hence, condition (2) of Proposition 2 holds and we can conclude, by Theorem 1, that ρ is an accepting run.

Given a set $\mathcal{F}$ of signed formulas, we define the sets $Pos(\mathcal{F})$ and $Neg(\mathcal{F})$ respectively as the sets of positive and negative propositions in $\mathcal{F}$, i.e. $Pos(\mathcal{F}) = \{p \in \mathcal{P} | \mathbf{T}p \in \mathcal{F}\}$, and $Neg(\mathcal{F}) = \{p \in \mathcal{P} | \mathbf{F}p \in \mathcal{F}\}$.

Theorem 3 *Let $\sigma \in \mathcal{L}(\mathcal{B}(\phi))$. Then there is a model $M = (\sigma, V)$ such that $M, \varepsilon \models \phi$.*

Proof Let ρ be an accepting run. for each $\tau \in prf(\sigma)$ let $\rho(\tau) = (\mathcal{F}_\tau, x_\tau, f_\tau)$. The model $M = (\sigma, V)$ can be obtained by defining $V(\tau) \in 2^{\mathcal{P}}$ such that $V(\tau) \supseteq Pos(\mathcal{F}_\tau)$ and $V(\tau) \cap Neg(\mathcal{F}_\tau) = \emptyset$.

It is easy to prove by induction on the structure of formulas that, for each τ and for each formula α, if $\mathbf{T}\alpha \in \mathcal{F}_\tau$ then $M, \tau \models \alpha$, and if $\mathbf{F}\alpha \in \mathcal{F}_\tau$ then $M, \tau \not\models \alpha$. In particular, for until formulas labelled **T** we make use of Theorem 1 and of Proposition 2, case 2, while for until formulas labelled **F** we make use of Proposition 2, case 3. From $\mathbf{T}\phi \in \mathcal{F}_\epsilon$, it follows that $M, \epsilon \models \phi$.

3.4. Complexity

It is known that for $\pi \in Prg(\Sigma)$, we can construct in polynomial time a non-deterministic finite state automaton $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = [[\pi]]$ such that the number of states of $\mathcal{A}$ is linear in the size of π [7]. The expansion of each until formula $\alpha \mathcal{U}^{\mathcal{A}(q)} \beta$ in the initial formula ϕ introduces at most a number of formulas which is linear in the size of $\mathcal{A}$ and, hence, is linear in the size of π. In fact, observe that the expansion of the until formula $\alpha \mathcal{U}^{\mathcal{A}(q)} \beta$ (and its descendants) introduces at most $|Q_\mathcal{A}|$ subformulas of the form $\alpha \mathcal{U}^{\mathcal{A}(q')} \beta$, with $q' \in Q_\mathcal{A}$. Let $\alpha_1 \mathcal{U}^{\pi_1} \beta_1$,....,$\alpha_n \mathcal{U}^{\pi_n} \beta_n$ be all the until formulas occurring in the initial formula ϕ. It must be that $|\pi_1| + \ldots + |\pi_n| \leq |\phi|$. Hence, the number of until formulas which are introduced in the construction of the automaton is linear in the size of the initial formula ϕ. Therefore, in the worst case, the number of states of the Büchi automaton is exponential in the size of $|\phi|$.

4. Conclusions

In this paper we have presented a tableau-based algorithm for constructing a Büchi automaton from a $DLTL$ formula. The formula is satisfiable if the language recognized by the automaton is nonempty. The construction of the states of the automaton can be done on-the-fly during the search that checks for emptiness. As in [6] we make use of finite automata to verify the fulfillment of until formulas. However, the construction of the automaton given in [6] is based on the idea of generating all the (maximally consistent) sets of the subformulas of the initial formula. Moreover, rather then introducing the states of the finite automata in the global states of the Büchi automaton, we stay closer to the standard construction for LTL [2] and we detect the point of fulfillment of the until formulas by associating a finite automaton with each until formula (rather than a regular expression) and by keeping track of the evolution of the state of these (finite) automata during the expansion of temporal formulas.

This construction could be improved in various ways, in particular by adopting the techniques presented in [1].

5. Acknowledgements

This research has been partially supported by the project MIUR PRIN 2003 "Logic-based development and verification of multi-agent systems".

References

[1] M. Daniele, F. Giunchiglia and M.Y. Vardi. Improved automata generation for linear temporal logic. In *Proc. 11th CAV*, Springer LNCS vol. 1633, pp. 249–260, July 1999.

[2] R. Gerth, D. Peled, M.Y. Vardi and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proc. 15th work. Protocol Specification, Testing and Verification*, Warsaw, June 1995.

[3] L.Giordano, A.Martelli, and C.Schwind. Reasoning about actions in dynamic linear time temporal logic. *Logic Journal of the IGPL*, 9(2):289–303, 2001.

[4] L. Giordano, A. Martelli, and C. Schwind. Specifying and Verifying Systems of Communicating Agents in a Temporal Action Logic. In *Proc. AI*IA'03*, Pisa, Springer LNCS vol. 2829, pp. 262–274, September 2003.

[5] J.G. Henriksen and P.S. Thiagarajan. A Product Version of Dynamic Linear Time Temporal Logic. In *CONCUR'97*, 1997.

[6] J.G. Henriksen and P.S. Thiagarajan. Dynamic Linear Time Temporal Logic. In *Annals of Pure and Applied logic*, vol.96, n.1-3, pp.187–207, 1999.

[7] J. Hromkovic, S. Seibert and T. Wilke. Translating Regular Expressions into Small ε-Free Nondeterministic Finite Automata. In *Proc. STACS'97*, Springer LNCS vol. 1200, pp. 55–66, 1997.

[8] F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proc. 12th CAV*, Springer LNCS vol. 1855, pp. 247–263, 2000.

[9] M. Vardi and P. Wolper. Reasoning about infinite computations. In *Information and Computation* 115,1–37 (1994).

[10] P. Wolper. Temporal logic can be more expressive. In *Information and Control* 56,72–99 (1983).

[11] P. Wolper. Constructing Automata from Temporal Logic Formulas: A Tutorial. In *Proc. FMPA 2000*, Springer LNCS vol. 2090, pp. 261–277, July 2000.