A compositional hierarchical monitoring automaton construction for LTL

Deepak D'Souza and Raj Mohan Matteplackel

Department of Computer Science and Automation
Indian Institute of Science
Bangalore, India.

Abstract. In this paper we give a compositional (or inductive) construction of monitoring automata for LTL formulas. Our construction is similar in spirit to the compositional construction of Kesten and Pnueli [5]. We introduce the notion of hierarchical Büchi automata and phrase our constructions in the framework of these automata. We give detailed constructions for all the principal LTL operators including past operators, along with proofs of correctness of the constructions.

1 Introduction

Linear-Time Temporal Logic (LTL) was proposed by Pnueli in [8] as a language for specifying temporal properties of program executions. It has since become a popular specification language with widespread use in practical verification.

In recent years, the most popular approach to solving the verification problem (or the so-called model-checking problem) for LTL is the automata-theoretic approach of Vardi and Wolper [9]. The central idea in this approach is the construction of a formula automaton $\mathcal{A}_\varphi$ for a given LTL formula φ, which is a Büchi automaton that accepts precisely the models of φ. Given a finite-state system model $\mathcal{T}$ and an LTL specification φ, one can check whether $\mathcal{T}$ satisfies φ (ie. whether all runs of $\mathcal{T}$ satisfy φ) by checking whether $\mathcal{T}$ and $\mathcal{A}_{\neg\varphi}$ have a joint accepting run. If they do have a joint accepting run, then we have a witness to the fact that $\mathcal{T}$ does not satisfy the formula, and if not we know that $\mathcal{T}$ does indeed satisfy the formula. This technique is implemented in popular explicit-state model-checking tools like Spin [4] as well as symbolic model-checking tools like Sal [2].

The Vardi-Wolper formula automaton construction is "monolithic" in that it directly (as against "compositionally") constructs an automaton for the given formula. While the transition relation of the automaton has a simple description, the set of valid atoms (which play the role of states) do not. In fact many techniques (for example [3]) focus on generating the set of valid next states efficiently. This can be an impediment to applying symbolic model checking. Further, the automaton can have a number of states that is exponential in the size of the given formula. Verifying systems with large state spaces and reasonable sized specifications in this "one-shot" technique is often a problem, with model-checking tools running out of memory.

In the face of this problem, an interesting "compositional" alternative for both deductive and algorithmic verification for LTL (and more generally for CTL*) was proposed by Kesten and Pnueli in [5]. Here the task of verifying whether $\mathcal{T}$ satisfies a temporal logic φ is broken up into several sub-tasks of verifying sub-formulas of φ. The key step in this approach is a compositional construction of a "monitoring" automaton (or a "temporal tester" as termed in [5]) for a given LTL formula. A monitoring automaton for a formula φ is similar to a formula automaton for φ, except that it "monitors" the truth of φ at *all* points along an accepting run of the automaton on any candidate model for φ. The technique is compositional in that the monitoring automaton for a formula of the form $\psi U \eta$ is constructed purely from monitoring automata for ψ and η. The monitoring automaton is constructed as a composition of component automata, where the communication between component automata is through shared state variables, in a manner similar to modelling languages like Z [10] and Event-B[1]. The monitoring automaton so constructed is linear in the size of the given LTL formula, though of course the explicit Büchi automaton corresponding to it may have an exponential number of states. This linear-sized representation is also conducive for symbolic model-checking.

In this paper, our focus is on a compositional construction of a monitoring automaton for an LTL formula. We present such a construction which is similar in spirit to that of Kesten and Pnueli, though in the framework of Hierarchical Büchi Automata (HBA's) which we introduce for this purpose. HBA's are similar to synchronous products of Büchi automata, except that component automata can have "edge guards" which may disallow certain joint transitions in the product. Thus the communication in these automata is only through edge guards.

Our work differs from Kesten and Pnueli in several ways. Our constructions are more concise than those in [5]. In general our constructions can be seen to use an optimal number of states and transitions. We formalize the notion of "monitoring" in terms of universal and unambiguous Büchi automata, and explicitly show that our constructions conform to this restriction. Unlike [5], which do not address the issue of correctness of their constructions, our constructions are accompanied by detailed proofs of correctness.

Finally our framework of HBA's is useful in generalising the compositional construction to timed temporal logics like MITL [7]. Such a generalisation has been done earlier in [6] though in the setting of signals (rather than timed words, for which we find HBA's more convenient).

The rest of this paper is organised as follows. We begin with prelimary definitions in the next section, and then introduce hierarchical Büchi automata in Section 3. In Section 4 we define the notion of a monitoring automaton for LTL formulas, and go on to give our inductive constructions for monintoring automata in Section 5. In the following section we discuss the optimality of our construction. We finally close with a conclusion section.

2 Preliminaries

For a finite alphabet of symbols Σ, an *infinite* word over Σ is an infinite sequence of symbols from Σ. We denote the set of infinite words over Σ by Σ^ω and the empty word by ϵ. We use standard notations for regular expressions denoting languages over finite and infinite words. We use $\mathbb{N}$ for the set $\{0, 1, \ldots\}$ of natural numbers.

Let us fix an alphabet Σ for the rest of this paper. The syntax of an LTL formula over Σ is given by:

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \bigcirc\varphi \mid \ominus\varphi \mid \varphi \vee \varphi \mid \varphi U \varphi \mid \varphi S \varphi,$$

where $a \in \Sigma$. The natural interpretation of LTL is over words in Σ^ω. Let $\sigma \in \Sigma^\omega$ be a word of the form $a_0 a_1 \cdots$. Let $i \in \mathbb{N}$. Then the satisfaction relation $\sigma, i \models \varphi$ is given by:

$$\begin{array}{lll}
\sigma, i \models \top & & \\
\sigma, i \models a & \text{iff} & a_i = a \\
\sigma, i \models \neg\psi & \text{iff} & \sigma, i \not\models \psi \\
\sigma, i \models \psi \vee \eta & \text{iff} & \sigma, i \models \psi \text{ or } \sigma, i \models \eta \\
\sigma, i \models \bigcirc\psi & \text{iff} & \sigma, i+1 \models \psi \\
\sigma, i \models \ominus\psi & \text{iff} & i > 0 \text{ and } \sigma, i-1 \models \psi \\
\sigma, i \models \psi U \eta & \text{iff} & \exists k \geq i : \sigma, k \models \eta, \text{ and } \forall j : i \leq j < k, \ \sigma, j \models \psi \\
\sigma, i \models \psi S \eta & \text{iff} & \exists k \leq i : \sigma, k \models \eta, \text{ and } \forall j : k < j \leq i, \ \sigma, j \models \psi.
\end{array}$$

We say that a word σ satisfies the LTL formula φ, written $\sigma \models \varphi$, if and only if $\sigma, 0 \models \varphi$, and set $L(\varphi) = \{\sigma \in \Sigma^\omega \mid \sigma \models \varphi\}$.

A *Buchi automaton* $\mathcal{B}$ is structure of the form (Q, Σ, S, E, F) where Q is a finite set of states, Σ is an alphabet, $S \subseteq Q$ is the set of initial states, $E \subseteq Q \times \Sigma \times Q$ is the set of transitions (or edges), and $F \subseteq Q$ is the set of final states.

Let $\sigma = a_0 a_1 \cdots$ be an infinite word over Σ. Then a *run* of $\mathcal{B}$ over σ is a map $\rho : \mathbb{N} \to Q$ satisfying the following conditions:

1. $q_0 \in S$.
2. For each i, $(\rho(i), a_i, \rho(i+1)) \in E$.

We call the run ρ *accepting* if $\rho(i) \in F$ for infinitely many i. In other words an accepting run visits a set of final states infinitely often. We say a word σ is accepted by $\mathcal{B}$ if $\mathcal{B}$ has an accepting run over σ. We define the language accepted by $\mathcal{B}$, denoted $L(\mathcal{B})$, to be the set of all words which are accepted by $\mathcal{B}$. We say $\mathcal{B}$ is *universal* if $L(\mathcal{B}) = \Sigma^\omega$ and *unambiguous* if $\mathcal{B}$ has at most one *accepting* run over each in Σ^ω.

It will be convenient to use a variant of Buchi automata in which the edges are marked initial and final. We call these *edge Buchi automata*. Formally, an *edge Buchi automaton* $\mathcal{B}$ is a structure of the form (Q, Σ, S, E, F) where Q is a finite set of states, Σ is an alphabet, $S \subseteq E$ is the set of *initial edges*, $E \subseteq Q \times \Sigma \times Q$ is the edge set, and $F \subseteq E$ is the set of *final edges*.

Let $\sigma = a_0 a_1 \cdots$ be an infinite word over Σ. Then a run of $\mathcal{B}$ over σ is a sequence of edges $\rho : \mathbb{N} \to E$ satisfying the following properties:

1. $\rho(0) \in S$.
2. There exist states $q_0 q_1 \cdots$ such that for each i, $\rho(i) = (q_i, a_i, q_{i+1})$.

We call the run ρ *accepting* if $\rho(i) \in F$ for infinitely many i. In other words an accepting run visits a set of final edges infinitely often.

Fig. 1 shows an example edge Buchi automaton $\mathcal{B}$ with two states. In the figure and the others that follow we adopt the following conventions. We write the action labels of the edges after the name label. Absence of an action label indicate that the edges can be taken on any action in Σ. The initial edges will be given as bold edges and the non-initial edges will be given as non-bold edges. For the final edges we use normal edges (non-dashed) and for the non-final edges we use dashed edges. The automaton has two initial edges labelled f_1 and f_4 and two final edges labelled f_1 and f_3. The edges f_1 and f_4 are enabled when the action a occurs and f_3 enabled when a b occurs. The edge f_2 is enabled on every action in Σ. The automaton accepts the language $(a^+(a+b)^*b)^\omega$.

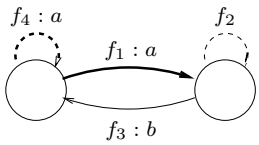

Fig. 1. An example edge Buchi automaton $\mathcal{B}$ accepting the language $(a^+(a+b)^*b)^\omega$. Bold face indicates initial edges, and full (as against dashed) edges indicate final edges.

It is not difficult to see that for a Buchi automaton one can construct a language-equivalent edge Buchi automaton and vice-versa. So these classes of automata are expressively equivalent.

3 Hierarchical Buchi automata

We now introduce the notion of hierarchical Buchi automata which we will use in our constructions for LTL. Let $L = [\mathcal{B}_n, \ldots, \mathcal{B}_1]$ be a list of edge Buchi automata, where $\mathcal{B}_i = (Q_i, \Sigma, E_i, S_i, F_i)$ for each i. Then we define the syntax of an *edge guard* w.r.t. L as follows:

$$g ::= \top \mid \mathcal{B}_i.e \mid \neg g \mid g \vee g \mid g \wedge g,$$

where $e \in E_i$. We denote the set of all edge guards w.r.t. L by $\mathcal{E}(L)$.

The semantics of an edge guard is defined as follows: an edge guard is evaluated over a joint transition of $\mathcal{B}_n, \ldots, \mathcal{B}_1$. Let $(e_n, \ldots, e_1)$, where $e_i \in E_i$, be a joint transition of $\mathcal{B}_n, \ldots, \mathcal{B}_1$ and let $g \in \mathcal{E}(L)$. Then we define the satisfaction relation, $(e_n, \ldots, e_1) \models g$, inductively as follows:

– $(e_n, \ldots, e_1) \models \top$.
– $(e_n, \ldots, e_1) \models \mathcal{B}_i.e$ iff $e_i = e$.
– Boolean combinations are handled in the expected manner.

A *hierarchical Buchi automaton* (or HBA) over Σ is a structure of the form $[\mathcal{C}_n, \ldots, \mathcal{C}_1]$ where each $\mathcal{C}_i$ is of the form $(\mathcal{B}_i, G_i)$ with $\mathcal{B}_i = (Q_i, \Sigma, S_i, E_i, F_i)$ an edge Buchi automaton and $G_i : E_i \to \mathcal{E}([\mathcal{B}_i, \ldots, \mathcal{B}_1])$ a labelling of edges of $\mathcal{B}_i$ with "level i" edge guards.

Let $\sigma \in \Sigma^\omega$. Then a run of $\mathcal{H}$ over σ is a joint run of $\mathcal{B}_n, \ldots, \mathcal{B}_1$ except that at each position the edge guards have to be satisfied. Formally, a run ρ of $\mathcal{H}$ over σ is a tuple $(\rho_n, \ldots, \rho_1)$ satisfying the following two conditions:

1. Each ρ_i is a run of $\mathcal{B}_i$ on σ.
2. For all $j \in \mathbb{N}$, $(\rho_n(j), \ldots, \rho_1(j)) \models \bigwedge_{i=1}^{n} G_i(\rho_i(j))$.

For a run $\rho = (\rho_n, \ldots, \rho_1)$ over $\mathcal{H}$ and a point $i \in \mathbb{N}$ we henceforth use $\rho(i)$ to mean the tuple $(\rho_n(i), \ldots, \rho_1(i))$.

The run ρ of $\mathcal{H}$ on σ is called *accepting* if each ρ_i is an accepting run of $\mathcal{B}_i$ on σ. A word σ is accepted by $\mathcal{H}$ if $\mathcal{H}$ has an accepting run over it. We define $L(\mathcal{H})$, the language accepted by $\mathcal{H}$, to be the set of all words which are accepted by $\mathcal{H}$. We say $\mathcal{H}$ is *universal* if $L(\mathcal{H}) = \Sigma^\omega$ and *unambiguous* if $\mathcal{H}$ has at most one accepting run for every word in Σ^ω.

Example 1. Fig. 2 shows the HBA $\mathcal{H}_1 = [\mathcal{C}_2, \mathcal{C}_1]$ over the alphabet $\{a, b\}$. It accepts the language, $L(\mathcal{H}_1) = \{b^i a^\omega \mid i \in \mathbb{N}\}$. In the figure the guards $\mathcal{C}_1.e_1$ and $\mathcal{C}_1.e_2$ of the automaton $\mathcal{C}_2$ refer to the edges e_1 and e_2 in the automaton $\mathcal{C}_1$. By convention we write the edge guards after the action labels of the edges, if any.

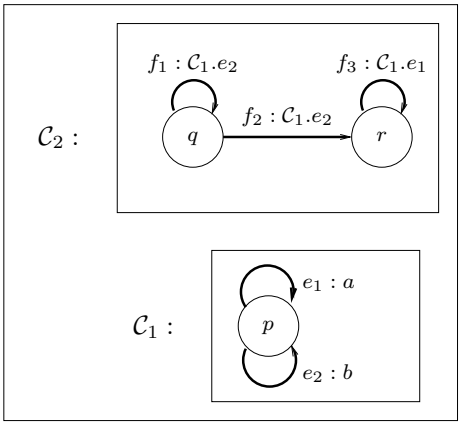

Fig. 2. HBA $\mathcal{H}_1$.

We now show that the class of ω-languages accepted by edge Buchi automata and HBA coincide. Clearly every edge Buchi automaton is an HBA. To see that every HBA-definable language is Buchi-definable too, let $\mathcal{H}$ be an HBA as defined above. Then we define an edge Buchi automaton $\mathcal{B}_{\mathcal{H}}$ as follows: $\mathcal{B}_{\mathcal{H}}$ is essentially the product of $\mathcal{B}_n \ldots, \mathcal{B}_1$ except that some edges in the product are disallowed based on the edge guards present on the edges of these automata.

- $Q = Q_1 \times \cdots \times Q_n \times \{0, \ldots, n\}$.
- $S = S_1 \times \cdots \times S_n$.
- $E \subseteq Q \times \Sigma \times Q$ is the set of edges defined as follows: for each $a \in \Sigma$, an edge $((p_n, \ldots, p_1, l), a, (q_n, \ldots, q_1, m)) \in E$ iff for all $i \in \{1, \ldots, n\}$ there exists an $e_i \in E_i$ of the form (p_i, a, g_i, q_i) such that the following conditions are satisfied:
  - $(e_n, \ldots, e_1) \models \bigwedge_{i=1}^{n} G_i(e_i)$.
  - $m = \begin{cases} (i+1) \bmod (n) & \text{if } i \geq 1 \text{ and } e_i \in F_i. \\ 1 & \text{if } i = 0. \\ i & \text{otherwise.} \end{cases}$
- $F = \{((p_n, \ldots, p_1, 0), a, (q_n, \ldots, q_1, 1)) \in E \mid a \in \Sigma\}$ is the set of final edges.

It is not difficult to see that $L(\mathcal{B}_{\mathcal{H}}) = L(\mathcal{H})$.

Example 2. Fig. 3 shows the language equivalent edge Buchi automaton $\mathcal{B}_{\mathcal{H}_1}$ corresponding to the HBA $\mathcal{H}_1$ of Example. 1.

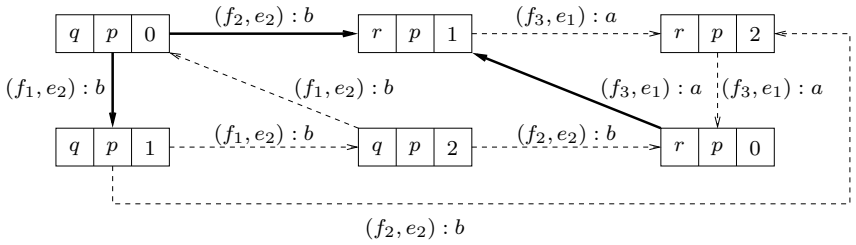

Fig. 3. Language-equivalent Buchi automaton $\mathcal{B}_{H_1}$ of the HBA $\mathcal{H}_1$.

4 Monitoring HBA for LTL

In this section we define the notion of a monitoring HBA for an LTL formula.

Let $\mathcal{H} = [C_n, \ldots, \mathcal{C}_1]$, where each $\mathcal{C}_i$ is of the form $(\mathcal{B}_i, G_i)$, be an HBA. Then by an *edge guard* over $\mathcal{H}$ we will mean an edge guard over the list of automata $\mathcal{B}_n, \ldots, \mathcal{B}_1$.

Definition 1. *Let φ be an* LTL *formula over Σ. Let $\mathcal{H}$ be an* HBA *over Σ and let g be an edge guard over $\mathcal{H}$. Then $(\mathcal{H}, g)$ is called a monitoring* HBA *for φ iff for every word $\sigma \in \Sigma^\omega$ the following conditions are satisfied.*

- *There exists a unique accepting run of $\mathcal{H}$ over σ. We denote this run by ρ_σ.*
- *The edge guard g monitors the truth of the formula along ρ_σ in the sense that $\sigma, i \models \varphi \iff \rho_\sigma(i) \models g$. We call g the monitoring guard for φ.*

We note that such an HBA $\mathcal{H}$ is necessarily unambiguous and universal.

The notion of a monitoring automaton for a formula is useful in verification because one can easily build a formula automaton for the formula from it. To see this consider an LTL formula φ. Let $(\mathcal{H}, g)$ be a monitoring automaton for φ and let $\mathcal{B}_\mathcal{H}$ be the language-equivalent edge Buchi automaton constructed as mentioned above. Let $\mathcal{C}$ be the automaton $\mathcal{B}_\mathcal{H}$ with the following restriction on its initial edges: an edge $(e_n, \ldots, e_1)$ of $\mathcal{C}$ is an initial edge if it is an initial edge in $\mathcal{B}_\mathcal{H}$ and $(e_n, \ldots, e_1) \models g$. It is not difficult to see that $L(\mathcal{C}) = L(\varphi)$, and hence $\mathcal{C}$ is a required formula automaton for φ.

5 Monitoring automaton construction for LTL

We now give a constructive proof showing that for any LTL formula φ we can construct a monitoring HBA for φ. In the proof, for an HBA $\mathcal{H} = [\mathcal{C}, \mathcal{C}_n, \ldots, \mathcal{C}_1]$ and a word $\sigma \in \Sigma^\omega$, if we are only interested in the run of $\mathcal{C}$ over σ then we use the compact notation (π, ρ) where π is a run of $\mathcal{C}$ over σ and ρ is a joint run of $\mathcal{C}_n, \ldots, \mathcal{C}_1$ over σ. And we also use the notation $(\pi(i), \rho(i))$ to refer to the edge tuple at position i in the run. For a list of edge Buchi automata $\mathcal{H} = [\mathcal{C}_n, \ldots, \mathcal{C}_1]$ and an edge Buchi automaton $\mathcal{C}$ we write $[\mathcal{C}, \mathcal{H}]$ to mean the list $[\mathcal{C}, \mathcal{C}_n, \ldots, \mathcal{C}_1]$.

In the figures that follow, the transitions are assumed to be labelled by Σ unless otherwise mentioned. Also, for convenience we write "ψ" instead of a monitoring guard for ψ.

Theorem 1. *Given an* LTL *formula φ we can effectively construct a monitoring* HBA *$(\mathcal{H}_\varphi, g_\varphi)$ for φ.*

Proof. We prove this theorem by induction on the structure of φ.

Case: $\varphi = a$, $a \in \Sigma$. For this case the monitoring HBA is $([\mathcal{C}_a], \mathcal{C}_a.e_a)$ where $\mathcal{C}_a$ is as shown in Fig. 4.

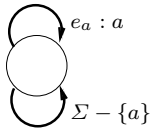

Fig. 4. Automaton $\mathcal{C}_a$.

Clearly the automaton $\mathcal{C}_a$ is deterministic and universal. Also, along the run of the automaton over a word σ, the guard e_a monitors the truth of the formula "a".

Case: $\varphi = \neg\psi$. By the induction hypothesis we have a monitoring HBA $(\mathcal{H}_\psi, g_\psi)$ for ψ. Then $(\mathcal{H}_\psi, \neg g_\psi)$ is a monitoring HBA for $\neg\psi$.

Case: $\varphi = \psi_1 \vee \psi_2$. Let $(\mathcal{H}_{\psi_1}, g_{\psi_1})$ and $(\mathcal{H}_{\psi_2}, g_{\psi_2})$ be the monitoring HBAs for ψ_1 and ψ_2 respectively. Then $([\mathcal{H}_{\psi_1}, \mathcal{H}_{\psi_2}], g_{\psi_1} \vee g_{\psi_2})$ is a monitoring HBA for φ.

Case: $\varphi = \bigcirc\psi$. By induction hypothesis we have a monitoring HBA $(\mathcal{H}_\psi, g_\psi)$ for ψ. Let $\mathcal{C}_{\bigcirc\psi}$ be the automaton shown in Fig. 5. Let $\mathcal{H}_{\bigcirc\psi} = [\mathcal{C}_{\bigcirc\psi}, \mathcal{H}_\psi]$ and $g_{\bigcirc\psi} = \mathcal{C}_{\bigcirc\psi}.(e_1 \vee e_2)$.

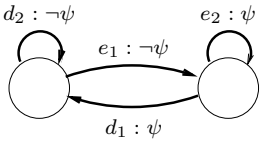

Fig. 5. Automaton $\mathcal{C}_{\bigcirc\psi}$.

We prove that $(\mathcal{H}_{\bigcirc\psi}, g_{\bigcirc\psi})$ is indeed a monitoring HBA for $\bigcirc\psi$. Let $\sigma \in \Sigma^\omega$ and let ρ be the unique accepting run of $\mathcal{H}_\psi$ over σ. Let π be a sequence of edges of $\mathcal{C}_{\bigcirc\psi}$ given by

$$\pi(i) = \begin{cases} e_1 \text{ if } \sigma, i \models \neg\psi \wedge \bigcirc\psi \\ e_2 \text{ if } \sigma, i \models \psi \wedge \bigcirc\psi \\ d_1 \text{ if } \sigma, i \models \psi \wedge \bigcirc\neg\psi \\ d_2 \text{ if } \sigma, i \models \neg\psi \wedge \bigcirc\neg\psi. \end{cases}$$

We now show that the edges in the sequence π connect up to form a valid run of $\mathcal{C}_{\bigcirc\psi}$ over σ. For example, consider the case when $\pi(i) = e_1$. Since $\pi(i) = e_1$ it must be the case that $\sigma, i+1 \models \psi$ and therefore $\pi(i+1)$ should be either e_2 or d_1, both of which are valid successor edges of e_1. So the transition relation of $\mathcal{C}_{\bigcirc\psi}$ is respected by π at position i. Other cases can be handled similarly. Finally, since all the edges of $\mathcal{C}_{\bigcirc\psi}$ are initial π is a valid run of $\mathcal{C}_{\bigcirc\psi}$ and since all the edges final too π is an accepting run of the automaton.

We now prove that (π, ρ) is an accepting run of $\mathcal{H}_{\bigcirc\psi}$ over σ. For this we need to show that the edge guards along π are satisfied at each position i. Consider the case when $\pi(i) = e_1$ once again. Then by the construction of π it must be the case that $\sigma, i \models \neg\psi$. Since ρ is the accepting run of $\mathcal{H}_\psi$ over σ, by the definition of a monitoring automaton $\rho, i \models \neg g_\psi$ and therefore $(\pi(i), \rho(i)) \models \neg g_\psi$. So the edge guard $\neg\psi$ is satisfied at position i in the run (π, ρ). Other cases can be handled similarly and we conclude (π, ρ) is an accepting run of $\mathcal{H}_{\bigcirc\psi}$.

Now, it remains to prove that (π, ρ) is the *only* accepting run of $\mathcal{H}_{\bigcirc\psi}$ over σ. Consider any accepting run of $\mathcal{H}_{\bigcirc\psi}$ over σ. Then it must be the form (π', ρ).

We argue that $\pi' = \pi$. Consider any position i, and suppose $\pi'(i) = e_1$. Since the edge guard on e_1 must be satisfied at $(\pi'(i), \rho(i))$ we have that $\rho(i) \models \neg g_\psi$ and $\rho(i+1) \models g_\psi$, i.e. $\sigma, i \not\models \psi$ and $\sigma, i+1 \models \psi$ By the construction of π, $\pi(i)$ should be e_1. Other cases can be handled similarly and we conclude $\pi' = \pi$.

Next we need to show that the guard g_φ monitors the truth of $\bigcirc\psi$ over (π, ρ). Consider any position i. Suppose $\sigma, i \models \bigcirc\psi$. Then:

$$\begin{aligned} &\sigma, i \models \bigcirc\psi \iff \sigma, i+1 \models \psi \\ &\iff \pi(i+1) = e_2 \text{ or } d_1 \\ &\iff \pi(i) = e_1 \text{ or } e_2. \end{aligned}$$

Case: $\varphi = \psi_1 U \psi_2$. Let $(\mathcal{H}_{\psi_1}, g_{\psi_1})$ and $(\mathcal{H}_{\psi_2}, g_{\psi_2})$ be the monitoring HBA for ψ_1 and ψ_2 respectively (which exist by induction). Let $\mathcal{C}_U$ be the automaton shown in Fig. 6. Let $\mathcal{H}_\varphi = [\mathcal{C}_U, \mathcal{H}_{\psi_2}, \mathcal{H}_{\psi_1}]$ and $g_\varphi = \mathcal{C}_U(e_1 \vee e_2 \vee e_3)$.

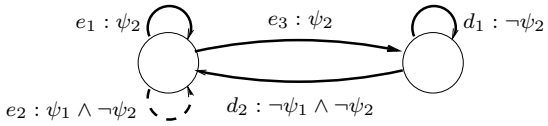

Fig. 6. Automaton $\mathcal{C}_U$.

We first prove that for any word $\sigma \in \Sigma^\omega$, $\mathcal{C}_U$ has an accepting run over it. Towards that end let us define a sequence of edges π of $\mathcal{C}_U$ as follows: for all $i \geq 0$ we define:

$$\pi(i) = \begin{cases} e_1 & \text{if } \sigma, i \models \psi_2 \text{ and } \sigma, i+1 \models \psi_1 U \psi_2 \\ e_2 & \text{if } \sigma, i \models (\psi_1 \wedge \neg\psi_2) \wedge (\psi_1 U \psi_2) \\ e_3 & \text{if } \sigma, i \models \psi_2 \text{ and } \sigma, i+1 \not\models \psi_1 U \psi_2 \\ d_1 & \text{if } \sigma, i \models \neg\psi_2 \text{ and } \sigma, i+1 \not\models \psi_1 U \psi_2 \\ d_2 & \text{if } \sigma, i \models \neg\psi_1 \wedge \neg\psi_2 \text{ and } \sigma, i+1 \models \psi_1 U \psi_2. \end{cases}$$

We now prove that π is an accepting run of $\mathcal{C}_U$ over σ. Observe that as all the edges in $\mathcal{C}_U$ are initial the edge $\pi(0)$ is initial. Now to show that the consecution relation is respected by the run at every position consider a position i in π. Suppose, if $\pi(i) = e_2$ then by the condition associated with e_2 we have that $\sigma, i \models \psi_1 U \psi_2$. This implies either $\sigma, i+1 \models \psi_1 U \psi_2$ or $\sigma, i+1 \models \psi_2$. Clearly $\pi(i+1)$ cannot be d_2 because if it were then neither $\psi_1 U \psi_2$ nor ψ_2 is true in σ at $i+1$. If $\psi(i+1) = d_1$ then by construction we have that $\sigma, i+1 \models \neg\psi_2 \wedge \neg(\psi_1 U \psi_2)$. This contradictions the assumption that $\sigma, i+1 \models \psi_2 \vee (\psi_1 U \psi_2)$. Thus the edge at position $i+1$ in π must be e_1, e_2 or e_3 all of which are valid consecutions e_2. Other cases can be handled similarly and we conclude that π valid run of $\mathcal{C}_U$.

To see that π is an accepting run of $\mathcal{C}_U$ note the only non-final edge is e_3. So if $\pi(i) = e_2$ for some i then it must be the case that there exists a $j > i$ such that $\sigma, j \models \psi_2$ which by the construction implies that $\pi(j)$ should be either e_1

or e_3. Thus every e_2 edge is followed by an e_1 or an e_3 both of which are final in π and therefore π accepting. This completes the proof of the claim that π is an accepting run of $\mathcal{C}_U$ over σ.

We now construct an accepting run of $\mathcal{H}_\varphi$ on σ using π. By the induction hypothesis there exist unique accepting runs ρ_1 and ρ_2 of $\mathcal{H}_{\psi_1}$ and $\mathcal{H}_{\psi_2}$ respectively, along which g_{ψ_1} and g_{ψ_2} monitor the truth of the formula ψ_1 and ψ_2. Consider the run (π, ρ), where ρ is the joint run ρ_1 and ρ_2. To show that (π, ρ) is an accepting run of $\mathcal{H}_\varphi$ on σ it is sufficient to show that the external guards of π are satisfied in (π, ρ). Again taking e_2 as an example, suppose $\pi(i) = e_2$. Then by the construction π, we know that ψ_1 and $\neg\psi_2$ are true at i in σ. Since g_{ψ_1} and g_{ψ_2} monitor ψ_1 and ψ_2 along ρ, it must be the case that $\rho, i \models g_{\psi_1} \wedge \neg g_{\psi_2}$. Hence the edge guard of e_2 is satisfied in ρ.

Next we show that (π, ρ) is the unique accepting run of $\mathcal{H}_\varphi$ over σ. Suppose it were not. Then there exists an accepting run π' of $\mathcal{C}_U$ such that (π', ρ) is an accepting run of $\mathcal{H}_U$. Let i be the first position such that $\pi(i) \neq \pi'(i)$. From the construction of $\mathcal{C}_U$ we observe that for any run the only plausible non-deterministic choices are between the edges e_1 and e_3, and between the edges d_1 and d_2. So let us assume that $\pi(i) = e_1$ and $\pi'(i) = e_3$. Then by the construction of π we have that $\sigma, i+1 \models \psi_1 U \psi_2$. Since π' is an accepting run it must be the case that $\pi'(i+1) = d_1$ or d_2. In either case we have that $\sigma, i+1 \not\models \psi_1 U \psi_2$, a contradiction. Similarly we can handle the other cases and we conclude that (π, ρ) is the unique accepting run of $\mathcal{H}_U$ over σ.

To prove that g_φ monitors the formula $\psi_1 U \psi_2$, consider a position i. Suppose $\sigma, i \models \varphi$. Then either $\sigma, i \models \psi_2$ in which case $\pi(i)$ should be either e_1 or e_3. Otherwise $\sigma, i \models \psi_1 \wedge \neg\psi_2$ and therefore $\pi(i) = e_2$. For the proof in the converse direction, if $\pi(i)$ is e_1 or e_3 then $\sigma, i \models \psi_2$ and if $\pi(i)$ is e_2 then $\sigma, i \models \psi_1 U \psi_2$.

Case: $\varphi = \ominus\psi$. Let $(\mathcal{H}_\psi, g_\psi)$ be a monitoring HBA for ψ. Let $\mathcal{C}_{\ominus\psi}$ be the automaton shown in Fig. 7 and let $\mathcal{H}_{\ominus\psi} = [\mathcal{C}_{\ominus\psi}, \mathcal{H}_\psi]$. It is not hard to see that if $\mathcal{H}_{\psi_1}$ is unambiguous and universal so is $\mathcal{H}_{\ominus\psi}$. It is also easy to verify that $\sigma, i \models \ominus\psi \iff \rho(i) \models \mathcal{C}_{\ominus\psi}.(e_1 \vee e_2)$ where ρ is the unique accepting run of $\mathcal{H}_{\bigcirc\psi}$ over σ. Therefore $(\mathcal{H}_{\ominus\psi}, \mathcal{C}_{\ominus\psi}.(e_1 \vee e_2))$ is monitoring HBA for $\ominus\psi$.

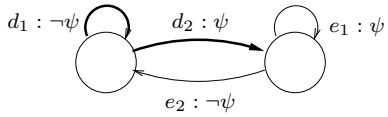

Fig. 7. Automaton $\mathcal{C}_{\ominus\psi}$.

Case: $\varphi = \psi_1 S \psi_2$. Let $(\mathcal{H}_{\psi_1}, g_{\psi_1})$ and $(\mathcal{H}_{\psi_2}, g_{\psi_2})$ be monitoring HBA for ψ_1 and ψ_2 respectively. Let $\mathcal{C}_S$ be the automaton shown in Fig. 8 and let $\mathcal{H}_S = [\mathcal{C}_S, \mathcal{H}_{\psi_2}, \mathcal{H}_{\psi_1}]$. We observe that as both the HBA $\mathcal{H}_{\psi_1}$ and $\mathcal{H}_{\psi_2}$ are unambiguous and universal by induction, $\mathcal{H}_S$ also is unambiguous and universal. Once again, it is easy to verify that $\sigma, i \models \psi_1 S \psi_2 \iff \rho(i) \models \mathcal{C}_S.(e_1 \vee e_2)$ where ρ is the

unique accepting run $\mathcal{H}_S$ on σ. Therefore $(\mathcal{H}_S, \mathcal{C}_S.(e_1 \vee e_2))$ is monitoring HBA for $\psi_1 S \psi_2$.

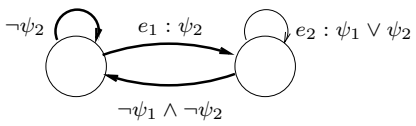

Fig. 8. Automaton $\mathcal{C}_S$.

□

Example 3. Fig. 9 shows an HBA $\mathcal{H}_U = [\mathcal{C}_U, \mathcal{C}_{\bigcirc b}, \mathcal{C}_b, \mathcal{C}_a]$ which along with the monitoring guard $\mathcal{C}_U.(e_1 \vee e_2 \vee e_3)$ monitors the LTL formula $aU(\bigcirc b)$.

We note that size of our monitoring automaton is linear in the size of the given LTL formula. More precisely the number of states in the monitoring HBA is bounded by $2 \cdot len(\varphi)$ and number of edges is bounded by $5 \cdot len(\varphi)$. Though the size of the induced Büchi automaton may still be $2^{O(len(\varphi))}$ in the worst case (as in the Vardi-Wolper automaton), the HBA is a linear-sized representation that is conducive to symbolic model-checking techniques.

6 Optimality of our constructions

We now prove that in general our constructions are optimal in terms of the number of states as well as edges used. To be precise, we prove that the component $\mathcal{C}_U$ of the monitoring automaton for a formula aUb, where $a, b \in \Sigma$, uses an optimal number of states and edges.

First let us argue that to monitor the formula aUb we require at least two states. Suppose there exists an automaton $\mathcal{A}$ with a single state such that $(\mathcal{A}, g)$ monitors aUb. Consider the accepting runs $\rho_1 = e_0 e_1 e_2 \cdots$ over the word $\sigma_1 = aaba^\omega$ and $\rho_2 = d_0 d_1 d_2 \cdots$ over $\sigma_2 = a^\omega$ of the automaton. Clearly $\sigma_1, 0 \models aUb$ and $\sigma_2, 0 \not\models aUb$. Since g monitors the truth of aUb we have that $\rho_1, 0 \models g$ and $\rho_2, 0 \not\models g$. This implies g in its most simplified form should be of the form $e_0 \vee g_1$, and should *not* be of the form $d_0 \vee g_2$, for some edge guards g_1 and g_2.

Let us now consider the run $\rho = e_0 d_1 d_2 \cdots$ of $\mathcal{A}$. It is not difficult that see that ρ is also an accepting run of $\mathcal{A}$ over σ_2. But this contradicts with the assumption that g monitors the formula as $\rho, 0 \models g$ and $\sigma_2, 0 \not\models aUb$.

In a similar way we can prove that in general our constructions for the remaining cases also use an optimal number of states.

To prove that any monitoring automaton for the formula aUb requires at least five edges let us assume, without loss of generality, that there exists an automaton $\mathcal{A}$ with four edges and a guard g over $\mathcal{A}$ such that $(\mathcal{A}, g)$ is a monitoring automaton for the formula. Let us first handle the case when, among the four edges, the formula is true when two of them say e_1 and e_2 are taken.

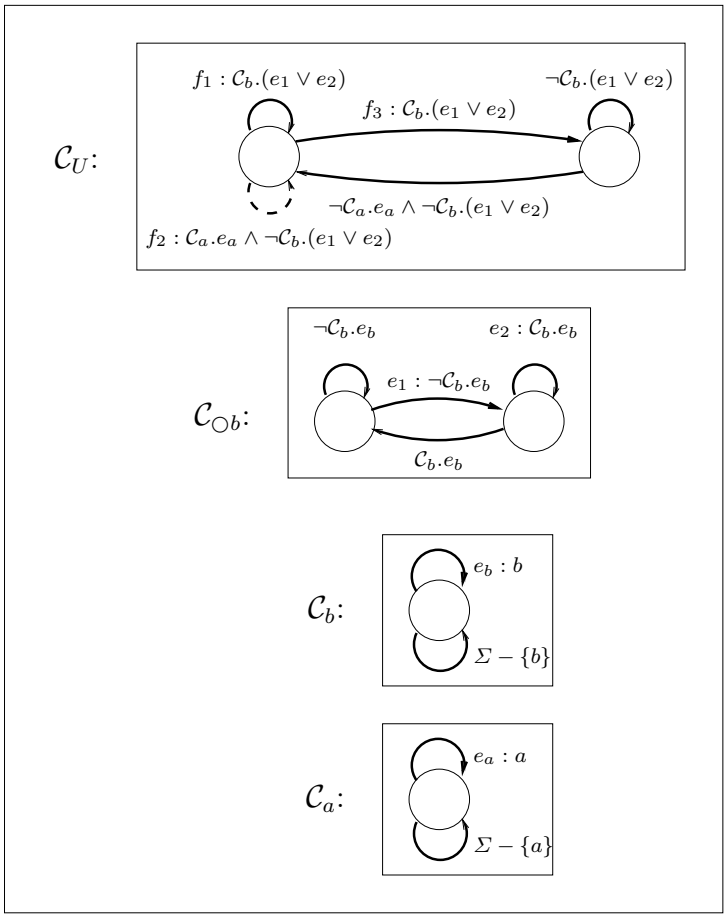

Fig. 9. HBA $\mathcal{H}_U$ for the formula $aU(\bigcirc b)$.

Now, consider an accepting run $\rho = f_0 f_1 \cdots$ of $\mathcal{A}$ over the word $\sigma = (ab)^\omega$. We first prove that there exists infinitely many i such that $f_i = e_1$ and infinitely many j such that $f_j = e_2$. Suppose that there exists a k such that for all $i \geq k$ $f_i = e_1$. Since ρ is accepting it must be the case that e_1 is final. Now consider the run $f_l f_{l+1} \ldots$ where $l > k$ and is even. It is not difficult to see that this is a valid accepting run of $\mathcal{A}$ over $(ab)^\omega$. But this implies that $\mathcal{A}$ has multiple accepting runs over σ, a contradiction to the assumption that $\mathcal{A}$ is unambiguous. Similarly we can handle the other case.

To prove that for all even i the edge $f_i = f_0$ (=e_2 say), suppose it were not. Then there exists an even position j such that $f_i \neq f_0$. Now consider the sequence of edges obtained from ρ by replacing the edge f_i with e_2. It is not difficult to show that this sequence is also an accepting run of $\mathcal{A}$ over σ, a contradiction as

$\mathcal{A}$ is unamiguous. Similarly we can prove that prove that for all odd i, $f_i = f_1$ ($=e_1$ say).

We also observe that the edge f_0 is non-accepting because if it were then $f_0 f_0 \cdots$ is an accepting run of $\mathcal{A}$ over the word a^ω along which g does not correctly monitor the formula. Further, we also deduce that the edge f_1 is final as ρ is an accepting run of $\mathcal{A}$. Since e_1 and e_2 are the only monitoring edges we can also conclude that both the edges e_1 and e_2 are from p to p for some state p.

We have proved above that any monitoring automaton for aUb requires at least 2 states. Since there are only two more edges, say d_1 and d_2, other than e_1 and e_2 it must be the case that d_1 is from state p to another state q and d_2 from q to p. Fig. 10 shows the edge transitions of $\mathcal{A}$.

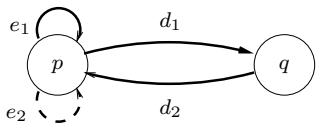

Fig. 10. Automaton $\mathcal{A}$.

It is not difficult to verify that this automaton cannot monitor the formula on any of its run over the word $(cccb)^\omega$, where c is another action in Σ. We can handle other cases similarly and hence we conclude that $(\mathcal{A}, g)$ is not a monitoring automaton for aUb.

Once again, on a similar note we can prove that in general our constructions for the remaining cases also use an optimal number of edges.

We would like to point out here that the monitoring automaton for the formula aUb (where a and b are actions) in the setting of Kesten and Pnueli [5] has has 8 states (of which 3 are dead states) and 27 edges. In contrast our monitoring automaton for the formula has 2 states and 5 edges, which we argue is optimal.

7 Conclusion

In this paper we have given a compositional construction for a monitoring automaton for arbitrary LTL formulas, in the framework of hierarchical Büchi automata. As pointed out in the introduction we depart from the similar construction given by Kesten and Pnueli in [5] in several aspects.

In addition, we would like to mention that in our construction it is easy to see by inspection that for a pure past LTL formula, the monitoring automaton is deterministic.

We also believe that our construction is useful from a pedagogical point of view, as each inductive step is fairly simple and intuitive, with simple and direct automata constructions.

References

1. Jean-Raymond Abrial. *Modeling in Event-B - System and Software Engineering.* Cambridge University Press, 2010.
2. Leonardo Mendon**c**ca de Moura, Sam Owre, Harald Rueß, John M. Rushby, Natarajan Shankar, Maria Sorea, and Ashish Tiwari. SAL 2. In Rajeev Alur and Doron Peled, editors, *CAV*, volume 3114 of *Lecture Notes in Computer Science*, pages 496–500. Springer, 2004.
3. Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of Linear Temporal Logic. In Piotr Dembinski and Marek Sredniawa, editors, *PSTV*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1995.
4. Gerard J. Holzmann. The model checker SPIN. *IEEE Trans. Software Eng.*, 23(5):279–295, 1997.
5. Yonit Kesten and Amir Pnueli. A compositional approach to CTL* verification. *Theor. Comput. Sci.*, 331(2-3):397–428, 2005.
6. Oded Maler, Dejan Nickovic, and Amir Pnueli. From MITL to Timed Automata. In Eugene Asarin and Patricia Bouyer, editors, *FORMATS*, volume 4202 of *Lecture Notes in Computer Science*, pages 274–289. Springer, 2006.
7. Raj Mohan Matteplackel. *Automata Constructions and Decision Procedures for Real-Time Logics.* PhD thesis, CSA Department, Indian Institute of Science, 2012.
8. Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.
9. 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.
10. Jim Woodcock and Jim Davies. *Using Z: Specification, Refinement, and Proof.* Prentice-Hall, 1996.