From MTL to Deterministic Timed Automata

Dejan Nickovic[1⋆] and Nir Piterman[2⋆⋆]

[1] EPFL, Lausanne, Switzerland
[2] Imperial College London, London UK

Abstract. In this paper we propose a novel technique for constructing timed automata from properties expressed in the logic MTL, under bounded-variability assumptions. We handle full MTL and in particular do not impose bounds on the future temporal connectives. Our construction is based on separation of the continuous time monitoring of the input sequence and discrete predictions regarding the future. The separation of the continuous from the discrete allows us to further determinize our automata. This leads, for the first time, to a construction from full MTL to deterministic timed automata.

1 Introduction

Timed automata [2], automata equipped with clocks, have been studied extensively in recent years as they provide a rigorous model for reasoning about quantitative time. Together with other formalisms such as real-time logics, real-time process algebras and timed Petri nets, they constitute an underlying theoretical basis for the specification and verification of real-time systems. The main attraction of timed automata is due to their suitability for modeling certain time-dependent phenomena, and the decidability of their reachability (or empty language) problem, a fact that has been exploited in several verification tools, e.g. Kronos [21] and Uppaal [12]. Recently there has also been interest in timed games and synthesis of timed controllers (e.g., [6]).

As in the untimed case, we would like to combine the model of timed automata with a powerful logic. Many variants of real-time logics [11, 4, 9, 8] as well as timed regular expressions [5] have been proposed. However, unlike the untimed case, the correspondence between simply-defined logics and variants of timed automata is not simple. One of the most popular dense-time extensions of LTL is the logic MITL introduced in [3] as a restriction of the logic MTL [11]. The principal modality of MITL is the timed until $\mathcal{U}_I$ where I is some non-singular interval. A formula $p\mathcal{U}_{(a,b)}q$ is satisfied by a model at any time instant t that admits q at some $t_0 \in (t+a, t+b)$, and where p holds continuously from t to t_0. Decidability of MITL was established in [3] by converting an MITL formula to a nondeterministic timed automaton and analyzing the structure of

⋆ Supported in part by the EU COMBEST project. Part of this work was done while this author was at Verimag, CNRS, Grenoble, France.

⋆⋆ Supported in part by the UK EPSRC project *Complete and Efficient Checks for Branching-Time Abstractions* (EP/E028985/1). Part of this work was done while this author was a visiting researcher at Verimag, CNRS, Grenoble, France.

that automaton. Further investigations of MITL and MTL suggested alternative translations of MITL to nondeterministic timed automata [14, 15] and used alternating timed automata to show decidability of MTL in certain circumstances [17].

In many cases, such as synthesis of timed controllers or online monitoring of timed behavior, we are interested in translating temporal specifications to deterministic timed automata. For MITL, this is, unfortunately, impossible [13]. Consider, for example, the formula $\varphi = \Box_{(0,a)}(p \rightarrow \Diamond_{(a,b)} q)$, which says that for every $t \in (0, a)$, if p is true at time t then there is a time point $t' \in (t+a, t+b)$ in which q holds. In order to construct a deterministic automaton for φ, we need infinite memory to remember all occurrences of p within the interval $(0, a)$. Furthermore, timed automata cannot be determinized [2]. Even if a fragment of MITL can be recognized by deterministic timed automata, we cannot use the usual constructions for translation of MITL to timed automata. Indeed, then there is no way to further determinize the automaton. We have to come up with specialized constructions that go directly to deterministic timed automata.

This is the approach taken in [15]. In order to enable the translation to deterministic automata, the assumption of bounded variability is taken. That is, there is a bound on the number of changes in the input signal in a given time interval. As mentioned, we cannot take the normal translation to nondeterministic automata and then use determinization, we have to come up with specialized constructions that go directly to deterministic automata. This is especially problematic when the formula includes predictions about the future, namely, future temporal connectives. In [15], only the 'safety' fragment of MTL is considered. They consider invariance properties, where the invariant may include past temporal operators and bounded future operators. Instead of saying that the invariant holds at time t, we wait until the bounded future operators in φ have elapsed and then, looking back, we can decide if the invariant continues to hold. This is done by effectively converting bounded future into past. As the past is naturally deterministic [13], deterministic automata can be constructed directly from bounded MTL.

The construction in [15] distinguishes between two reasons for the impossibility of determinization for timed languages. The first is unbounded variability, a property that is used to show that timed automata cannot be determinized and complemented. The second is a-causality, the value of a formula at time t may depend on the value of input at time $t' > t$. By assuming bounded variability we eliminate the first reason. In [15] it is conjectured that a-causality on its own can be handled by 'normal' determinization. Here, we prove this conjecture by a construction that takes full MTL (under bounded variability) to nondeterministic timed automata followed by a determinization construction for timed automata.

What is the problem with determinization for the normal conversion of MTL to timed automata? Said constructions use clocks to accumulate regions where every subformula is true. When coming to determinize timed automata, one cannot collect all the possible values of the clocks associated with these parts[3].

We take a different approach, by separating our timed automata into two parts. The first is a 'normal' *deterministic* timed automaton that uses clocks to collect times of events. The second is a *dependent timed automaton*, an automaton that uses the clocks

[3] A notable exception is [17] reasoning about alternating timed automata with one clock; although the resulting structure is not a timed automaton.

controlled by the first part, to make *discrete* predictions regarding the future. It is the dependent timed automaton that we later determinize.

We identify an 'interest region' for the MTL formula. We construct property monitors, deterministic timed automata that memorize (using clocks) all events regarding propositions in this interest region. Decisions regarding truth values of subformulas are delayed and decided retroactively. Consider a bounded temporal property $\varphi = \psi_1 \mathcal{U}_{(a,b)} \psi_2$. Given that we 'know' the truth value of ψ_1 and ψ_2 in the memorized region, we can 'deduce' the truth value of φ in part of the region. Specifically, if the memorized region is $(t-f, t)$ (i.e., we are now reading time t and the memory region is of size f), then we deduce the truth value of φ in $(t-f, t-b)$. We do not add states for such subformulas. When considering an unbounded subformula $\varphi = \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$, we construct a small automaton, that makes guesses regarding the future, for a given time point within the 'knowledge' region.

Finally, based on two assumptions, we show that our dependent timed automata can be determinized. First, transitions cannot be enabled throughout the stay in a state. Second, when a transition is enabled the automaton can stay for a little while in the target state. The determinization construction is a slight variant of the determinization construction for normal finite automata on infinite words [20, 18].

Even when ignoring the option to determinize, our construction has many advantages when compared with previous constructions. In our construction the number of clocks depends on the number of propositions, the depth of future depth of the longest chain of *nested* temporal operators, and the bounded variability of the input. In previous constructions clocks are allocated according to variability and the depth of each operator separately. Thus, if operators are not nested within one another many more clocks may be required. Furthermore, in our construction the number of states associated with every unbounded until is constant while in previous construction every temporal operator requires states that are proportional to the number of active interval that temporal operator may have. Furthermore, we note that if we consider the fragment of bounded future operators considered in [15], then the automata produced by our construction are also going to be deterministic without applying an extra determinization step. Finally, existing translations usually require automata that alternate between states defined over "singular" (zero-duration) and "open" intervals, difficult to implement in current tools such as IF [7], Kronos [21], and Uppaal [12]. In our construction timed automata use only left-closed right-open intervals, easier to handle by existing tools.

2 Definition of MTL

A signal over a domain D is a function $w : \mathbb{T} \to D$ where $\mathbb{T}$ is the time domain. The time domain is either the set $\mathbb{R}_{\geq 0}$ of non-negative real numbers in the case of infinite-length signals or an interval $[0, r)$ if the signal is of finite length. We focus on the case where D is a finite domain, typically the set $\mathbb{B}^n$ of Boolean vectors over n variables (or propositions). We denote by w_p the projection of w to the proposition p. Concatenation of two finite signals w_1 and w_2 defined over $[0, r_1)$ and $[0, r_2)$, respectively, is the finite signal $w = w_1 \cdot w_2$, defined over $[0, r_1 + r_2)$ as $w[t] = w_1[t]$ for $t < r_1$ and $w[t] = w_2[t - r_1]$ for $t \geq r_1$.

We introduce the future fragment of MTL interpreted over dense-time signals. The syntax of MTL is defined by the grammar

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \ \mathcal{U}_I \ \varphi_2$$

where p belongs to a set $P = \{p_1, \ldots, p_n\}$ of propositions and I is an interval of the form $[b, b]$, (a, b), or (a, ∞) where $0 \leq a < b$ are integer numbers. We say that an interval I is *unbounded* if it is of the form (a, ∞), otherwise it is *bounded*. As in LTL the basic MTL operators can be used to derive other standard Boolean and temporal operators, in particular the time-constrained *eventually* $\Diamond_I \varphi = \mathrm{T} \ \mathcal{U}_I \ \varphi$ and *always* $\Box_I \varphi = \neg \Diamond_I \neg\varphi$. It is also possible to express other types of intervals such as $[a, b]$, $[a, b)$, $[a, \infty)$, etc.

The semantics of an MTL formula φ with respect to an n-dimensional Boolean signal w is described via the satisfiability relation $(w, t) \models \varphi$, indicating that the signal w satisfies φ at time t, according to the following recursive definition.

$$\begin{array}{lcl}
(w,t) \models p & \leftrightarrow & w_p[t] = 1 \\
(w,t) \models \neg\varphi & \leftrightarrow & (w,t) \not\models \varphi \\
(w,t) \models \varphi_1 \vee \varphi_2 & \leftrightarrow & (w,t) \models \varphi_1 \text{ or } (w,t) \models \varphi_2 \\
(w,t) \models \varphi_1 \ \mathcal{U}_I \ \varphi_2 & \leftrightarrow & \exists\, t' \in t + I \text{ st } (w,t') \models \varphi_2 \text{ and} \forall\, t'' \in (t,t')\ (w,t'') \models \varphi_1
\end{array}$$

A formula φ is satisfied by w if $(w, 0) \models \varphi$.

It is well known that MTL formulas can be translated to nondeterministic timed automata [3, 14]. In Section 4 we suggest a new construction for converting MTL to timed automata. The construction is based on computing a bound f from the formula and the automaton at time t memorizes the input signal at the interval $[t - f, t)$. The value of the formula is computed with a delay, that is, when the automaton is reading time t it computes the value of the formula for time $t - f$. In the remainder of this section we compute the size of the interval the automaton needs to memorize in order to make our construction work.

The truth value of a formula φ at time t depends on the input signal at some interval $[t, t + f)$. If φ does not contain subformulas with unbounded intervals then measuring this interval is straightforward. If φ does contain subformulas with unbounded intervals, then the truth value of φ at time t depends on guessing the future values of unbounded future operators. The main thing we note is that the guess regarding the future is Boolean – does the formula change its truth value in the future or not. For example, $\varphi = q\mathcal{U}_{(2,\infty)}r$ depends on the values of q in the interval $[t, t + 2 + 2\epsilon)$. [4] In case that q holds throughout the interval $(t, t + 2 + 2\epsilon)$, we guess whether q holds continuously until r starting at time $t + 2 + 2\epsilon$. Based on this guess, we have sufficient knowledge to establish whether φ holds at time t. The guess obviously needs to be checked. More formally, we define the function fut that determines the bound of this interval. We fix a

[4] Strictly speaking, the knowledge of the values of q in the interval $[t, t + 2 + \epsilon)$ is sufficient. The extra ϵ is necessary for determinization of timed automata obtained from MTL formulae.

small ϵ for the rest of the paper.

$\mathsf{fut}(p) = 0$ Where p is a proposition.
$\mathsf{fut}(\varphi_1 \vee \varphi_2) = max(\mathsf{fut}(\varphi_1), \mathsf{fut}(\varphi_2))$
$\mathsf{fut}(\neg\varphi_1) = \mathsf{fut}(\varphi_1)$
$\mathsf{fut}(\varphi_1 \mathcal{U}_I \varphi_2) = a + 2\epsilon + max(\mathsf{fut}(\varphi_1), \mathsf{fut}(\varphi_2))$ Where $I = (a, \infty)$.
$\mathsf{fut}(\varphi_1 \mathcal{U}_I \varphi_2) = b + max(\mathsf{fut}(\varphi_1), \mathsf{fut}(\varphi_2))$ Where $I = (a, b)$ or $I = [b, b]$.

The function fut is used also to decide where our 'certainty' regarding the truth value of a formula expires. For example, if we know the value of q and r in the interval $[t-4, t)$ our knowledge region for the formula $q\mathcal{U}_{(2,\infty)}r$ expires at time $t - 2 - 2\epsilon$. In case that q holds throughout the interval $(t-2-2\epsilon, t)$ then the truth value of $q\mathcal{U}_{(2,\infty)}r$ depends on a guess that q keeps holding until r becomes true. We do know, however, that $q\mathcal{U}_{(2,\infty)}r$ has not been falsified and may still hold at time $t - 2 - 2\epsilon$.

3 Timed Automata

We use a variant of timed automata that differs slightly from the classical definitions [2, 10, 1]. Our automata read multi-dimensional *dense-time* Boolean signals and output Boolean signals. Input and output are associated with states *and* sometimes transitions. We also extend the domain of clock values to include the special symbol $\perp$ indicating that the clock is currently *inactive* and extend the order relation on $\mathbb{R}_{\geq 0}$ accordingly by letting $\perp < v$ for every $v \in \mathbb{R}_{\geq 0}$. We freely use multiplication by -1 and comparison with negative values. It follows that $-\perp > -v$ for every $v \in \mathbb{R}_{\geq 0}$. For a set $A \subseteq \mathcal{R}^n$ we use $cl(A)$ to denote its closure (in the topological sense).

The set of valuations of a set $C = \{x_1, \ldots, x_n\}$ of clock variables, each denoted as $v = (v_1, \ldots, v_n)$, defines the clock space $\mathcal{H} = (\mathbb{R}_{\geq 0} \cup \{\perp\})^n$. A *configuration* of a timed automaton is a pair of the form (q, v) with q being a discrete state. For a clock valuation $v = (v_1, \ldots, v_n)$, $v + t$ is the valuation $(v'_1, \ldots, v'_n)$ such that $v'_i = v_i$ if $v_i = \perp$ and $v'_i = v_i + t$ otherwise. An *atomic clock constraint* is a condition of the form $x \bowtie y + d$ or $x \bowtie d$, where x and y are clocks, $\bowtie \in \{<, \leq, \geq, >\}$, and d is an integer. Let $\mathbb{A}(\mathcal{C})$ denote the set of atomic constraints over the set $\mathcal{C}$ of clocks. For a set X, let $\mathcal{B}^+(X)$ denote the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$). Let $\mathbb{C}(\mathcal{C}) = \mathcal{B}^+(\mathbb{A}(\mathcal{C}))$ denote the set of *constraints* over the set of clocks $\mathcal{C}$. We also view a constraint $c \in \mathbb{C}(\mathcal{C})$ as a subset $c \subseteq \mathcal{H}$. In what follows, we introduce free real variables to constraints and quantify over them. That is, we use constraints in the first-order theory of the reals where clocks in $\mathcal{C}$ are free variables. The elimination of quantifiers gives us constraints in $\mathbb{C}(\mathcal{C})$. We include a short discussion of quantifier elimination in Appendix A. We used the tool in [16] to eliminate quantifiers in some of the examples below.

Definition 1. *A* timed automaton *is $\mathcal{A} = \langle \Sigma, Q, \mathcal{C}, \lambda, I, \Delta, q_0, \mathcal{F} \rangle$, where Σ is the input alphabet, Q is a finite set of discrete states, and $\mathcal{C}$ is a set of clock variables. We assume that Σ is 2^{AP} for some set of propositions AP. We freely use Boolean combinations of propositions to denote sets of letters. The labeling function $\lambda : Q \to \Sigma$ associates an input letter with every state. The staying condition (invariant) I assigns to every*

state q a constraint $I(q) \in \mathbb{C}(\mathcal{C})$. The transition relation Δ consists of elements of the form (q, g, ρ, q') where q and q' are discrete states, the transition guard g is a subset of $\mathcal{H}$ defined by a clock constraint, and ρ is the update function, a transformation of $\mathcal{H}$ defined by an assignment of the form $x := 0$, $x := \perp$, or $x := y$ or a set of such assignments. Finally q_0 is the initial state. Transitions leaving q_0 have True *as their guard and can use only updates of the form $x := 0$. We consider generalized Büchi automata, where $\mathcal{F} \subseteq 2^Q$.*

The behavior of the automaton as it reads a signal w consists of a strict alternation between time progress periods, where the automaton stays in a state q as long as $w[t] = \lambda(q)$ and I_q holds, and discrete instantaneous transitions guarded by clock conditions. Formally, a step of the automaton is one of the following:

- A time step $(q, v) \xrightarrow{\sigma^t} (q, v+t)$ $t \in \mathbb{R}_{>0}$ such that $\sigma = \lambda(q)$ and $(v, v+t) \subseteq cl(I_q)$.
- A discrete step: $(q, v) \xrightarrow{\delta} (q', v')$, for some transition $\delta = (q, g, \rho, q') \in \Delta$, such that $v \in g$ and $v' = \rho(v)$.

Let $v_\perp = (\perp, \ldots, \perp)$ be the assignment of $\perp$ to all clocks. A *run* of the automaton starting from a configuration $(q_0, v_\perp)$ is a finite or infinite sequence of strictly alternating time and discrete steps of the form

$$\zeta : (q_0, v_0) \xrightarrow{\delta_0} (q_1, v_1) \xrightarrow{\sigma_1^{t_1}} (q_1, v_1 + t_1) \xrightarrow{\delta_1} (q_2, v_2) \xrightarrow{\sigma_2^{t_2}} (q_2, v_2 + t_2) \xrightarrow{\delta_2} \cdots,$$

such that $\Sigma_i t_i$ diverges. A run ζ is accepting if for every $F \in \mathcal{F}$ the set of times instances in which states from F are visited is unbounded. The input signal carried by the run is $\sigma_1^{t_1} \cdot \sigma_2^{t_2} \cdots$, where we abuse notation and denote by $\sigma_i^{t_i}$ the concatenation of the punctual signal σ_i and the open signal $\sigma_i^{t_i}$. That is $\sigma_i : [0, t_i) \to \Sigma$ such that forall $t \in [0, t_i)$ we have $w(t) = \sigma_i$.

Given two timed automata $A_i = \langle \Sigma_i, Q_i, \mathcal{C}_i, \lambda_i, I_i, \Delta_i, q_0^i, \mathcal{F}_i \rangle$, for $i \in \{1, 2\}$, their composition $A_1 \parallel A_2$ is $\langle \Sigma_1 \times \Sigma_2, Q_1 \times Q_2, \mathcal{C}_1 \cup \mathcal{C}_2, \lambda, I, \Delta, (q_0^1, q_0^2), \mathcal{F} \rangle$, where $\lambda(q_1, q_2) = (\lambda_1(q_1), \lambda_2(q_2))$, $I(q_1, q_2) = I_1(q_1) \wedge I_2(q_2)$, and $\mathcal{F} = \{S \times T \mid S \in \mathcal{F}_1 \text{ and } T = Q_2, \text{ or } S = Q_1 \text{ and } T \in \mathcal{F}_2\}$. The transition Δ includes three kinds of transitions as follows.

- *Simultaneous transitions* $((q_1, q_2), g, \rho, (q_1', q_2'))$, where $(q_i, g_i, \rho_i, q_i') \in \Delta_i$ for $i \in \{1, 2\}$, $g = g_1 \wedge g_2$ and $\rho = \rho_1 \cup \rho_2$,
- *Left-side transitions* $((q_1, q_2), g, \rho, (q_1', q_2))$, where $(q_1, g_1, \rho, q_1') \in \Delta_1$ and $g = g_1 \wedge I_2(q_2)$, and
- *Right-side transitions* $((q_1, q_2), g, \rho, (q_1, q_2'))$, where $(q_2, g_2, \rho, q_2') \in \Delta_2$ and $g = I_1(q_1) \wedge g_2$.

A timed automaton is deterministic if from every reachable configuration every event and every 'non-event' leads to exactly one configuration. This means that the automaton cannot make both a 'silent' transition and a time passage in the same configuration.

Definition 2. *A* deterministic timed automaton *is an automaton whose guards and staying conditions satisfy:*

1. *For every two distinct transitions (q, g_1, ρ_1, q_1) and (q, g_2, ρ_2, q_2) we have either $\lambda(q_1) \neq \lambda(q_2)$ or $g_1 \wedge g_2$ is unsatisfiable.*

2. *For every transition (q, g, ρ, q'), either $\lambda(q) \neq \lambda(q')$ or the intersection of g and $I(q)$ is either empty or isolated, i.e., there does not exist an open interval (t, t') such that $(t, t') \subseteq I(q)$ and $(t, t') \cap g \neq \emptyset$.*

We introduce dependent timed automata. These are automata that do not have clocks of their own, however can 'read' the clock values of other timed automata. Furthermore, we add to dependent timed automata output and the composition of dependent timed automata allows one automaton to read the output of the other. Dependent timed automata allow us to separate the continuous from the discrete when reasoning about MTL formulas. The composition of a dependent timed automaton with a timed automaton results in a timed automaton.

Definition 3. *A* dependent timed automaton *is $B = \langle \Sigma, \Gamma, Q, \mathcal{C}, \gamma, I, \Delta, q_0, \mathcal{F} \rangle$, where the following is different from timed automata. We add an output alphabet Γ, an output function $\gamma : Q \to \Gamma$, and remove the labeling function. The staying condition I assigns to every state a Boolean combination of atomic constrains and input letters $I : Q \to \mathcal{B}^+(\mathbb{A}(\mathcal{C}) \cup \Sigma)$. The transition relation Δ consists of elements of the form (q, g, o, q'), where $g \in \mathcal{B}^+(\mathbb{A}(\mathcal{C}) \cup \Sigma)$ is a Boolean combination of atomic constraints and input letters, $o \in \Gamma$ is an output, and the clock update is removed. We assume that $\Gamma = 2^{AP}$ for some set of proposition AP and we freely use propositions to define staying conditions and transition guards.*

Consider two dependent timed automata $B_i = \langle \Sigma_i, \Gamma_i, Q_i, \mathcal{C}, \gamma_i, I_i, \Delta_i, q_0^i, \mathcal{F}_i \rangle$ for $i \in \{1, 2\}$, where $\Sigma_2 = \Sigma_1 \times \Gamma_1$. The composition $B_1 \otimes B_2$, where B_2 reads the output of B_1, is the following dependent timed automaton. Let $B_1 \otimes B_2 = \langle \Sigma$, $\Gamma_1 \times \Gamma_2$, $Q_1 \times Q_2$, $\mathcal{C}$, γ, I, Δ, (q_0^1, q_0^2), $\mathcal{F} \rangle$, where $\gamma(q_1, q_2) = (\gamma_1(q_1), \gamma_2(q_2))$ and $\mathcal{F} = \{S \times T \mid S \in \mathcal{F}_1$ and $T = Q_2$, or $S = Q_1$ and $T \in \mathcal{F}_2\}$. The staying condition is $I(q_1, q_2) = I_1(q_1) \wedge simp(\gamma_1(q_1), I_2(q_2))$ where $simp(\gamma_1(q_1), \varphi)$ is the constraint obtained from φ by replacing $\gamma_1(q_1)$ by true and all other letters in Γ_1 by false. The transition Δ is similar to the composition of timed automata and includes (a) simultaneous transitions $((q_1, q_2), g, (o_1, o_2), (q_1', q_2'))$, where $(q_i, g_i, o_i, q_i') \in \Delta_i$ for $i \in \{1, 2\}$ and $g = g_1 \wedge simp(o_1, g_2)$, (b) left-side transitions $((q_1, q_2), g, (o_1, \gamma_2(q_2)), (q_1', q_2))$, where $(q_1, g_1, o_1, q_1') \in \Delta_1$ and $g = g_1 \wedge simp(o_1, \gamma_2(q_2))$, and (c) right-side transitions $((q_1, q_2), g, (\gamma_1(q_1), o_2), (q_1, q_2'))$, where $(q_2, g_2, o_2, q_2') \in \Delta_2$ and $g = I_1(q_1) \wedge simp(\gamma_1(q_1), g_2)$.

Consider a timed automaton $A_1 = \langle \Sigma_1, Q_1, \mathcal{C}, \lambda_1, I_1, \Delta_1, q_0^1, \mathcal{F}_1 \rangle$ and a dependent timed automaton $B_2 = \langle \Sigma_1, \Gamma_2, Q_2, \mathcal{C}, \gamma_2, I_2, \Delta_2, q_0^2, \mathcal{F}_2 \rangle$. Their composition $A_1 \otimes B_2$ is the timed automaton $A = \langle \Sigma, Q_1 \times Q_2, \mathcal{C}, \lambda, I, \Delta, (q_0^1, q_0^2), \mathcal{F} \rangle$, where $\lambda(q_1, q_2) = \lambda_1(q_1)$, and $\mathcal{F} = \{S \times T \mid S \in \mathcal{F}_1$ and $T = Q_2$, or $S = Q_1$ and $T \in \mathcal{F}_2\}$. The staying condition $I(q_1, q_2) = I_1(q_1) \wedge simp(\lambda_1(q_1), I_2(q_2))$. The transition Δ includes (a) simultaneous transitions $((q_1, q_2), g, \rho_1, (q_1', q_2'))$, where $(q_1, g_1, \rho_1, q_1') \in \Delta_1$, $(q_2, g_2, o_2, q_2') \in \Delta_2$, $g = g_1 \wedge app(\rho, simp(\lambda_1(q_1'), g_2))$, and $app(\rho, g_2)$ applies the effect of ρ on g_2, e.g., if ρ includes $x := y$ we replace x in g_2 by y, (b) left-sided transitions $((q_1, q_2), g, \rho_1, (q_1', q_2))$, where $(q_1, g_1, \rho_1, q_1') \in \Delta_1$ and $g = g_1 \wedge app(\rho_1, simp(\lambda_1(q_1'), \gamma_2(q_2)))$ (c) right-sided transitions $((q_1, q_2), g, \emptyset, (q_1, q_2'))$, where $(q_2, g_2, o_2, q_2') \in \Delta_2$ and $g = I_1(q_1) \wedge simp(\lambda_1(q_1), g_2)$. Notice that the composition

$$\Delta = \left\{ \begin{array}{l} (q_{in}, \mathit{True}, \emptyset, q_0), (q_{in}, \mathit{True}, x_1 := 0, q_1), \\ (q_{2i}, y_1 < f, x_{i+1} := 0, q_{2i+1}), (q_{2i+1}, y_1 < f, y_{i+1} := 0, q_{2i+2}), \\ \left(q_{2i+2}, y_1 = f, \left\{ \begin{array}{l} x_1 := x_2, y_1 := y_2, \ldots, x_i := x_{i+1}, y_i := y_{i+1}, \\ x_{i+1} := \bot, y_{i+1} := \bot \end{array} \right\}, q_{2i}\right), \\ \left(q_{2i+2}, y_1 = f, \left\{ \begin{array}{l} x_1 := x_2, y_1 := y_2, \ldots, x_i := x_{i+1}, y_i := y_{i+1}, \\ x_{i+1} := 0, y_{i+1} := \bot \end{array} \right\}, q_{2i+1}\right) \\ \left(q_{2i+3}, y_1 = f, \left\{ \begin{array}{l} x_1 := x_2, y_1 := y_2, \ldots, x_i := x_{i+1}, y_i := y_{i+1}, \\ x_{i+1} := \bot, y_{i+1} := \bot \end{array} \right\}, q_{2i+1}\right) \\ \left(q_{2i+3}, y_1 = f, \left\{ \begin{array}{l} x_1 := x_2, y_1 := y_2, \ldots, x_i := y_{i+1}, y_i := y_{i+1}, \\ x_{i+1} := x_{i+2}, y_{i+1} := 0, x_{i+2} := \bot \end{array} \right\}, q_{2i+2}\right) \end{array} \right\}$$

Fig. 1. The transition of proposition monitor.

of a timed automaton with a dependent timed automaton causes the output of the second to disappear. Thus, $A \otimes B_1 \otimes B_2$ should be read as $A \otimes (B_1 \otimes B_2)$.

4 From MTL to Nondeterministic Timed Automata

We suggest a novel construction for the conversion of MTL formulas to timed automata. The advantage of this construction is that it effectively distinguishes between discrete guesses relating to occurrences in the future (made by dependent timed automata) and the accumulation of knowledge with clocks (made by timed automata). This separation allows us to construct a deterministic automaton for the formula in Section 5. Note that the number of clocks depends on the structure of the formula through the computed bound fut and the construction of the proposition monitors. This section starts by introducing proposition monitors, deterministic timed automata that log information about the input. We then expose our construction using a simple example. Finally, we proceed to the formal construction that explains how to handle general MTL formulas.

We start by introducing a timed automaton that memorizes the times in which a proposition is true (cf. [15], event recorder Figure 2). Given a formula φ let $f = \mathsf{fut}(\varphi)$. The automaton is going to memorize all events occurring in the interval $[t-f, t)$. Let k be the number of changes possible in a proposition in 1 time unit. It follows that in the interval $[t-f, t)$ there can be at most $\lceil \frac{fk}{2} \rceil$ different sub-intervals in which the proposition is true. Thus, we need $2 \cdot \lceil \frac{fk}{2} \rceil$ clocks to memorize their start and end times. Let $n = \lceil \frac{fk}{2} \rceil$. Consider a proposition p. Let $\mathcal{A}_p = \langle 2^{\{p\}}, Q, \mathcal{C}, \lambda, I, \Delta, q_{in}, \{Q\} \rangle$, where $\mathcal{C} = \{x_1^p, \ldots, x_n^p, y_1^p, \ldots, y_n^p\}$, $Q = \{q_{in}, q_0, \ldots, q_{2n}\}$, $\lambda(q_{2i}) = \emptyset$, $\lambda(q_{2i+1}) = \{p\}$, for $j > 1$ we have $I(q_j) = y_1^a < f$ and $I(q_0) = I(q_1) = \mathit{True}$ and Δ is given in Figure 1. One such proposition monitor is given in Figure 2.

We now expose our construction through an example. Consider the formula $\varphi = \Box(p \mathcal{U} \Diamond_{(0,1)} q)$. It is simple to see that $\mathsf{fut}(\varphi) = 1 + 4\epsilon$. Suppose that every proposition changes at most k times during every 1 time unit. We build the automata $\mathcal{A}_p$ and $\mathcal{A}_q$. with bound $f = 1 + 4\epsilon$ and the bounded variability constant k. In addition, we construct a simple timed automaton $\mathcal{A}_z$ with one state and one clock z that measures the time since time 0. It is used to check whether the bound f has been reached. In

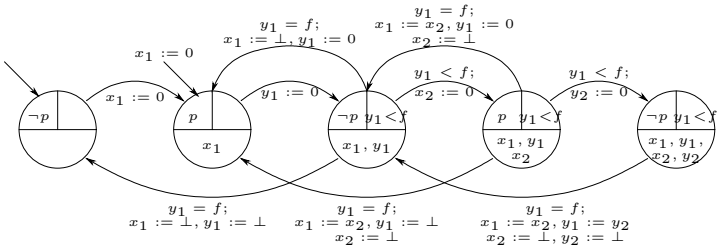

Fig. 2. Proposition monitor for p, where $f = \mathsf{future}(\varphi)$ and $\lceil \frac{fk}{2} \rceil = 2$.

what follows, we think about the current time point as 0. For example, if $x_1^r = 2.37$ and $y_1^r = 1.49$, from our point of view r was true during the interval $[-2.37, -1.49)$. In order to proceed with the construction of the dependent timed automata we first define constraints that describe the truth values of subformulas of φ.

- Consider the subformula $\varphi_1 = \Diamond_{(0,1)} q$. We construct a constraint $\varphi_1(t)$ that describes when the subformula φ_1 holds at time $t \in [-1-4\epsilon, -1)$. Thus, no states are created with respect to this subformula. The formula φ_1 is true at time $t \in [-1-4\epsilon, -1)$ if $\exists t' \in (t, t+1).q(t')$, that is if for some i we have $-x_i^q < t+1$ and $t < -y_i^q$.
  In general, for bounded subformulas we use the information already stored in the state space and clocks of timed automata and dependent timed automata to construct a constraint that tells us when the subformula holds.
- Consider the subformula $\varphi_2 = p\mathcal{U}\varphi_1$. For such an unbounded formula our construction includes two parts. First, we construct a dependent timed automaton that makes the guess regarding the future. Second, we use this dependent timed automaton to define constraints that tell us when the formula holds just like bounded formulas.
  - We start by constructing a dependent timed automaton for the truth value of φ_2 at exactly $-1-2\epsilon$. As φ_2 is an unbounded until, its truth value may depend on events that occur arbitrarily far in the future. The problematic situation is when p holds throughout $(-1-2\epsilon, -1-\epsilon)$ but φ_2 does not hold there. The dependent timed automaton for the value of φ_2 is given in Figure 3, where the invariants and guards are as follows.
    Intuitively, in state s_1 the automaton sees that p holds throughout the interval $(-1-2\epsilon, -1-\epsilon)$ and guesses that it will hold continuously until φ_1 holds. In state s_2 the automaton sees that φ_1 holds somewhere in $(-1-2\epsilon, -1-\epsilon)$ and p holds up to that point. In state s_3 the knowledge is just like in s_1, however, the automaton guesses that p falls before φ_1 rises. In state s_4 the automaton sees a violation to the until, that is, p falls before φ_1 rises within $(-1-2\epsilon, -1-\epsilon)$. Formally, the state invariants are as follows:

$$
\begin{aligned}
I_1, I_3 &: \forall t \in (-1-2\epsilon, -1-\epsilon) p(t) \wedge \neg\varphi_1(t) \\
I_2 \quad &: \exists t \in (-1-2\epsilon, -1-\epsilon).\varphi_1(t) \wedge \forall t' \in (-1-2\epsilon, t) p(t) \\
I_4 \quad &: \exists t \in (-1-2\epsilon, -1-\epsilon) \neg p(t) \wedge \forall t' \in (-1-2\epsilon, t).\neg\varphi_1(t)
\end{aligned}
$$

That is, I_1 and I_3 maintain that p is true throughout $(-1-2\epsilon, -1-\epsilon)$ but $\Diamond_{(0,1)}\, q$ is false. The invariant I_2 maintains that somewhere in $(-1-2\epsilon, -1-\epsilon)$ we have $\Diamond_{(0,1)}\, q$ holds and p holds up to that point and I_4 maintains that p falls before $\Diamond_{(0,1)}\, q$.

The transition guards use the extra ϵ to look ahead a bit further. Intuitively, a guard makes sure that it is possible to cross to the next state and uses the extra look-ahead to make sure that it is possible to *stay* in the next state a little while. This is required in order to be able to determinize the automaton. Formally, the transition guards are as follows:

$$\begin{aligned} g_1, g_3 &: I_1 \wedge \exists t \in [-1-\epsilon, -1).\forall t' \in [-1-\epsilon, t).p(t') \wedge \neg\varphi_1(t') \\ g_2 &: I_2 \vee (I_1 \wedge \exists t \in (-1-2\epsilon, -1-\epsilon).\forall t' \in (-1-2\epsilon, t). \\ &\qquad \exists t'' \in (t', t'+\epsilon).\varphi_1(t'') \wedge \forall t''' \in (t', t'').p(t''') \\ g_4 &: I_4 \vee (I_3 \wedge \exists t \in (-1-2\epsilon, -1-\epsilon).\forall t' \in (-1-2\epsilon, t). \\ &\qquad \exists t'' \in (t', t'+\epsilon).\neg p(t'') \wedge \forall t''' \in (t', t'').\neg\varphi_1(t''') \end{aligned}$$

In case that determinization is not pursued, the extra look-ahead can be removed. In this case the guards are I_1 for transitions into s_1 and s_3, $I_2 \vee I_1$ for transitions into s_2, and $I_4 \vee I_1$ for transitions into s_4.

The output of the automaton is the proposition p_{φ_2}. The states s_1 and s_2 as well as all the transitions entering them are labeled by p_{φ_2}. All other states and transitions are labeled by $\neg p_{\varphi_2}$. The only unfair state is s_1, where we promise to fulfill the until in the future. The automaton waits until the clock z reaches $1+4\epsilon$, when the memory 'fills up', and only then starts working.

- The dependant timed automaton and the proposition p_{φ_2} express the truth value of φ_2 at exactly $-1-2\epsilon$. We now describe the constraint that matches the truth value of φ_2 for every $t \in [-1-4\epsilon, -1-2\epsilon]$. Formally, for every $t \in [-1-4\epsilon, -1-2\epsilon]$ we have φ_2 holds at t if one of the following holds.
  - If $t = -1-2\epsilon$ and p_{φ_2} holds.
  - If $t < -1-2\epsilon$, p_{φ_2} holds, and forall $t' \in (t, -1-2\epsilon]$, $p(t')$ holds.
  - If $t < -1-2\epsilon$ and there exists $t' \in (t, -1-2\epsilon)$ such that φ_1 holds at t' and forall $t'' \in (t, t')$ we have $p(t)$.

  Let $\varphi_2(t)$ be the constraint obtained by eliminating quantifiers from this disjunction.

– We now proceed to the top formula containing φ_2, using the proposition p_{φ_2}. We construct a dependent timed automaton for the value of $\varphi = \Box\,\varphi_2$ at time $-1-4\epsilon$. We construct a dependent timed automaton with two states s_0 and s_3 with the following transitions and invariants. The invariant of s_0 is $z < 1+4\epsilon$. The invariant of s_3 is $\forall t \in (-1-4\epsilon, -1-3\epsilon).\varphi_2(t)$. The unique transition is (s_0, g, p_φ, s_3), where g is $z = 1+4\epsilon \wedge \exists t' \geq -1-3\epsilon.\forall t'' \in [-1-3\epsilon, t').\varphi_2(t'')$. The state s_3 is an accepting state and both the transition and s_3 are labeled by p_φ.

Recall, that $\Box\,\varphi_2 \equiv \neg(\mathit{True}\,\mathcal{U}\neg\varphi_2)$. It follows that the dependent timed automaton for φ can be thought of as a copy of the automaton in Figure 3. The invariant I_4 is *false* making state s_4 redundant. Furthermore, as $\Box$ is the top most operator in φ, we enforce the truth of φ by enabling only the transition from s_0 to s_3, making states s_1 and s_2 unreachable.

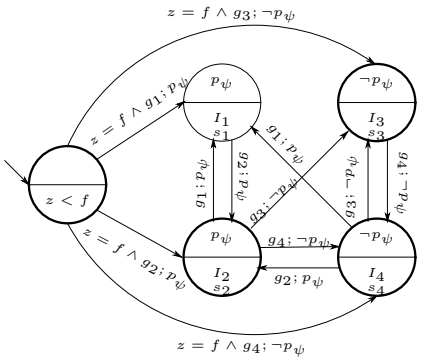

Fig. 3. A dependent timed automaton for unbounded until.

This completes the construction of the timed automaton for the example.

We now turn to the general construction. Consider an MTL formula φ. For subformulas ψ of φ, we construct constraints that say when ψ holds at time t. For a bounded subformula there is no need to add states. For an unbounded subformula ψ (i.e., Until where the upper limit is ∞) we add a small dependent timed automaton that computes the value of ψ at the time point $\mathsf{fut}(\psi) - \mathsf{fut}(\varphi)$. Based on this dependent timed automaton we then compute the constraint that says when ψ holds at time t. We start with constructing property monitors for all the propositions appearing in the formula. For a subformula ψ, we construct by induction for $t \in [-f, -\mathsf{fut}(\psi))$ the constraint $\psi(t)$.

- For a proposition p and for $t \in [-f, 0)$, the constraint $p(t)$ is $\exists i. -x_i^p \leq t \wedge -y_i^p > t$. Notice that the quantification on i depends on the number of clocks in the proposition monitors.
- For subformulas of the form $\neg\psi$, $\psi_1 \vee \psi_2$, or $\psi_1 \wedge \psi_2$ the combination of the constraints is straightforward. The range allowed for t is the minimal range allowed by ψ_1 and ψ_2.
- Consider a subformula $\psi = \psi_1 \mathcal{U}_I \psi_2$, where $I = (a, b)$ or $I = [b, b]$.
  By definition $\mathsf{fut}(\psi) = b + max(\mathsf{fut}(\psi_1), \mathsf{fut}(\psi_2))$. It follows that ψ_1 is defined in $[-f, -\mathsf{fut}(\psi_1))$ and ψ_2 is defined in $[-f, -\mathsf{fut}(\psi_2))$. So, for $t \in [-f, -\mathsf{fut}(\psi))$ it is always the case that $t + b$ is in the range where $\psi_1(t)$ and $\psi_2(t)$ are defined. In the case that $I = (a, b)$, for $t \in [-f, -\mathsf{fut}(\psi))$ we set $\psi(t) = \exists t' \in (t + a, t + b). \psi_2(t') \wedge \forall t'' \in (t, t'). \psi_1(t'')$. In the case that $I = [b, b]$, for $t \in [-f, -\mathsf{fut}(\psi))$ we set $\psi(t) = \psi_2(t + b) \wedge \forall t' \in (t, t + b). \psi_1(t')$.
- Consider a formula $\psi = \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.
  By definition $\mathsf{fut}(\psi) = a + 2\epsilon + max(\mathsf{fut}(\psi_1), \mathsf{fut}(\psi_2))$. It follows that ψ_1 is defined in $[-f, -\mathsf{fut}(\psi_1))$ and ψ_2 is defined in $[-f, -\mathsf{fut}(\psi_2))$. So, for $t \in [-f, -\mathsf{fut}(\psi)]$ it is always the case that $(t, t + a + 2\epsilon)$ is contained in the range where ψ_1 and ψ_2 are defined. We first construct a dependent timed automaton for ψ and then use the output of this dependent timed automaton for computing a constraint for ψ.
  - We construct a dependent timed automaton for the truth value of ψ at time $t = -\mathsf{fut}(\psi)$. Again, we use the automaton in Figure 3, where the guards and

the invariants are as follows.

$$
\begin{aligned}
I_1, I_3 &: \forall t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + a + \epsilon).\psi_1(t) \quad \wedge \\
&\quad \forall t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + \epsilon).\neg\psi_2(t) \\
I_2 &: \exists t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + \epsilon). \\
&\qquad \psi_2(t) \wedge \forall t' \in (-\mathsf{fut}(\psi), t).\psi_1(t) \\
I_4 &: \exists t(-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + a + \epsilon). \\
&\qquad \neg\psi_1(t) \ \wedge \forall t' \in (-\mathsf{fut}(\psi) + a, t).\neg\psi_2(t) \\
g_1, g_3 &: I_1 \wedge \exists t \in [-\mathsf{fut}(\psi) + a + \epsilon, -\mathsf{fut}(\psi) + a + 2\epsilon). \\
&\qquad \forall t' \in [-\mathsf{fut}(\psi) + a + \epsilon, t).\psi_1(t') \wedge \neg\psi_2(t') \\
g_2 &: I_2 \vee (I_1 \wedge \exists t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + \epsilon).\forall t' \in (-\mathsf{fut}(\psi), t). \\
&\quad \exists t'' \in (t' + a, t' + a + \epsilon).\psi_2(t'') \wedge \forall t''' \in (t, t'')\psi_1(t''')) \\
g_4 &: I_4 \vee (I_3 \wedge \exists t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + \epsilon).\forall t' \in (-\mathsf{fut}(\psi), t). \\
&\quad \exists t'' \in (t', t' + a + \epsilon).\neg\psi_1(t'') \wedge \forall t''' \in (t' + a, t'')\neg\psi_2(t''')
\end{aligned}
$$

States s_1, s_2 and their incoming transitions are labeled by p_ψ, all other states and transitions by $\neg p_\psi$, and s_1 is the only unfair state.

- We now construct the constraint that describes the truth value of ψ. For every $t \in [-\mathsf{fut}(\varphi), -\mathsf{fut}(\psi)]$ we set $\psi(t)$ to the disjunction of:
  - $t = -\mathsf{fut}(\psi) \wedge p_\psi$,
  - $t < -\mathsf{fut}(\psi) \wedge p_\psi \wedge \forall t' \in (t, -\mathsf{fut}(\psi)].\psi_1(t)$, or
  - $t < -\mathsf{fut}(\psi) \wedge \exists t' \in (t + a, -\mathsf{fut}(\psi)).\psi_2(t') \wedge \forall t'' \in (t, t').\psi_1(t)$.

This completes the inductive part of the construction.

Consider the case that the top-most connective is $\mathcal{U}$. Then from the initial state of the dependent timed automaton associated with $\mathcal{U}$ we allow only the transitions to s_1 and s_2. Dually, if the top most connective is $\neg\mathcal{U}$ we allow only the transitions to s_3 and s_4. In case that the top-most connective is not $\mathcal{U}$, we add a dependent timed automaton with three states. Let Γ be the alphabet $2^{AP'}$ where AP' are all the propositions introduced during the construction. Let $B = \langle \Sigma \times \Gamma, \{o\}, \{q_0, q_1, q_2\}, \mathcal{C}, \gamma, I, \Delta, q_0, \{\{q_1\}\}\rangle$, where $\gamma(q_0) = \gamma(q_1) = \gamma(q_2) = o$, $I(q_0) = z < f$, and $I(q_1) = I(q_2) = \mathit{True}$. The transition relation is $\Delta = \{(q_0, g_1, o, q_1), (q_0, g_2, o, q_2)\}$, where $g_1 = \varphi(-f) \wedge z = f$ and $g_2 = \neg\varphi(-f) \wedge z = f$. That is, the dependent timed automaton B enters state q_1 if φ is true at time 0 and state q_2 if φ is false at time 0. State q_1 is an accepting sink state and state q_2 is a rejecting sink state.

Finally, the timed automaton for φ, denoted by A_φ, is the composition of the proposition monitors with the dependent timed automata constructed above. The proof of the following theorem is given in Appendix B.

Theorem 1. *For every timed sequence w we have $w \models \varphi$ iff $w \in A_\varphi$.*

Corollary 1. *For every MTL formula φ with m propositions, n unbounded temporal operators, and inputs of bounded variability k, there exists a nondeterministic timed automaton with $2mk \cdot \mathsf{fut}(\varphi) + 1$ clocks and $3(2k \cdot \mathsf{fut}(\varphi))^m 4^n$ states that accepts the language of φ.*

Proof. Every proposition monitor has $2k \cdot \mathsf{fut}(\varphi)$ clocks. There is an additional clock for measuring the time elapsed since 0.

Every proposition monitor has $2k \cdot \mathtt{fut}(\varphi)$ states. Every unbounded temporal operator has at most four states. The automaton associated with the top level formula φ has at most three states. ⌟

5 Deterministic Timed Automata

In this section we show that the automata constructed in Section 4 can be determinized. This part is based on the separation of the real domain and the clocks from the automata constructed for the discrete guesses. We give additional requirements on the structure of dependent timed automata that enable us to further determinize them. With these additional conditions in place, we can apply (a variant of) the subset construction to determinize dependent timed automata. In order to simplify presentation and save space, we do not present the more complicated constructions derived from determinization for automata on infinite words. Determinization for automata on infinite words uses the subset construction to follow all the runs of the nondeterministic automaton simultaneously. Then, an extra gadget is added to ensure that one of the runs followed by the subset construction is accepting. Adding this extra gadget on top of the subset construction in our case is straight forward.

We assume that in our dependent timed automata for every transition (q, g, o, q') the intersection of g and $I(q)$ is either empty or isolated, i.e., there does not exist an open interval (t, t') such that $(t, t') \subseteq I(q)$ and $(t, t') \cap g \neq \emptyset$ and if (q, g, o, q') is enabled at time t then there is a small interval (t, t') such that $(t, t') \subseteq I(q')$. It is simple to see that the dependent timed automata constructed in Section 4 satisfy this condition.

Let $B = \langle \Sigma, \Gamma, Q, \mathcal{C}, \gamma, I, \Delta, q_0, \mathcal{F} \rangle$ be a dependent timed automaton satisfying these conditions. We are going to construct a deterministic dependent timed automaton D that follows simultaneously all the runs of B. The construction is based on the subset construction [19]. Thus, every state of D is a set of states of B. In a state $Q' \subseteq Q$, the automaton D is following a set of runs of B ending in the states Q'. For a set $Q' \subseteq Q$ let $I(Q')$ be $\bigwedge_{q \in Q'} I(q)$ and $\overline{I}(Q')$ be $\bigwedge_{q \in Q'} \neg I(q)$. Let $\Delta(Q') = \{(q, g, o, q') \in \Delta \mid q \in Q'\}$. For a set $\Delta' \subseteq \Delta$ let $g(\Delta')$ be $\bigwedge_{(q,g,o,q') \in \Delta'} g$ and $\overline{g}(\Delta')$ be $\bigwedge_{(q,g,o,q') \in \Delta'} \neg g$.

Given a set of runs ending in states in Q' how can these runs be extended? Some runs are extended by staying in the same state, some runs are extended by crossing discrete transitions (and cannot be extended by crossing other discrete transitions), and some runs cannot be extended. We represent such a choice by a set $T \subseteq Q' \cup \Delta(Q')$. Let $stay(T) = T \cap Q'$ be the states whose runs are extended by staying in the same state. In particular, all transitions from $stay(T)$ have to be disabled. Let $\Delta(T) = \Delta(Q') \cap T$ be the discrete transitions taken by states in Q. Let $deadend(Q', T)$ be the set of states $q \in Q'$ such that $q \notin T$ and for every $(q', g, o, q'') \in T$ we have $q' \neq q$. That is, the states whose runs cannot be extended. Let $move(T) = Q' \setminus (stay(T) \cup deadend(T))$, the set of states that have some transitions going out of them in T. Let $target(T)$ be $stay(T) \cup \{q \mid \exists (q', g, o, q) \in trans(T)\}$. Our deterministic automaton is going to take a T-transition from a set associated with Q. The guard, $g(T)$ of this transition is the conjunction of the following: (a) $I(stay(T)) \wedge \overline{g}(\Delta(stay(T)))$ – the invariant of states whose run is extended must be satisfied and the guards of the transitions exiting them must not be satisfied, (b) $g(trans(T)) \wedge \overline{g}(\Delta(move(T)) \setminus T)$ – the guards of transitions

that are crossed must be satisfied and the guards of transitions that are not crossed must not be satisfied, and (c) $\overline{I}(deadend(Q', T)) \wedge \overline{g}(\Delta(deadend(Q', T)))$ – the invariant of states whose run is going to end must not be satisfied and the guards of the transitions exiting them must not be satisfied. The case that $deadend(T)$ and $trans(T)$ are empty corresponds to all runs being extended and is not interesting.

We construct the 'deterministic' dependent timed automaton D. The differences from the construction in [19] are in bold. Let $D = \langle \Sigma, \{o\}, S, \mathcal{C}, \gamma', I', \Delta', s_0, \alpha \rangle$, where $S = 2^Q$, $s_0 = \{q_0\}$, the acceptance condition α is ignored, **for every state $s \in S$ we have $\gamma'(s) = o$ and $I'(s) = I(s)$**, and for every state $s \in S$ and **every set $T \subseteq s \cup \Delta(s)$ we add to Δ' the transition $(s, g(T), o, target(T))$.**

Theorem 2. *For every deterministic timed automaton $A = \langle \Sigma, R, \mathcal{C}, \lambda, I, \Delta, r_0, \mathcal{F} \rangle$, we have $A \otimes D$ is deterministic.*

Proof. Consider two transitions $\delta_1 = ((r, s), g_1, \rho_1, (r_1, s_1))$ and $\delta_2 = ((r, s), g_2, \rho_2, (r_2, s_2))$. If at least one of them is the result of a simultaneous transition or a left-side transition (wlog δ_1) then clearly either $g_1 \wedge g_2$ is unsatisfiable or $\lambda(r_1) \neq \lambda(r_2)$ by the definition of deterministic timed automaton. Suppose that both are right-side transitions. Thus, $r = r_1 = r_2$ and $\rho_1 = \rho_2 = \emptyset$. Let $T_1, T_2 \subseteq set(s) \cup \Delta(set(s))$ be the sets used for the transitions (s, g_1', o, s_1) and (s, g_2', o, s_2) in D. Either $stay(T)$, $move(T)$, $trans(T)$, or $deadend(T)$ is different. In each case, the guard of T_1 includes some conjunct c such that $\neg c$ is a conjunct in the guard of T_2.

Consider a transition $\delta = ((r, s), g, \rho, (r', s'))$. If δ is the result of a simultaneous transition or a left-side transition then by A being deterministic, either $\lambda(r) \neq \lambda(r')$ or the intersection of g and $I(r)$ is either empty or isolated. Consider the case that δ is the result of a left-side transition. Let $T \subseteq set(s) \cup \Delta(set(s))$ be the set used for definition of (s, g', o, s') in D. Either $trans(T)$ or $deadend(T)$ is not empty. If $trans(T)$ is not empty, then there is a state $q \in set(s)$ and a transition (q, g, o, q') in T. By assumption g is enabled in an isolated point in $I(q)$ and δ is enabled at most in an isolated point in $I(s)$. If $deadend(T)$ is not empty, then there is a state $q \in set(s)$ such that $I(s)$ includes $I(q)$ and g includes $\neg I(q)$. Hence, $I(s) \wedge g$ is unsatisfiable. ∎

In the full version, we include the timed version of [18] and show that the resulting deterministic automaton accepts the same language.

6 Conclusions

Motivated by practical problems such as synthesis of timed controllers from real-time temporal specifications, we developed a procedure that translates full MTL to deterministic timed automata. Apart from its practical applications, our construction provides a better understanding of sources of non-determinism associated with real-time temporal logics and timed automata.

In the future, we intend to investigate further improvements of our construction:

– In the full version consider MTL with past operators. This extension does not increase the complexity of the construction as satisfaction of past operators depends only on the observation of memorized events in the proposition monitors.

– Interpret the logic over finite signals, in the context of online monitoring of timed behaviors.
– Optimize and improve the translation. One straightforward improvement would require a smarter memorization of events in the proposition monitors.
– Implement the translation presented in this paper, in order to investigate its applicability to the practical controller synthesis from MTL specifications.

Acknowledgments

We thank J. Harrison and B. Cook for their explanations regarding quantifier elimination.

References

1. R. Alur. Timed automata. In *CAV*, LNCS 1633, pages 8–22. Springer, 1999.
2. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126(2):183–236, 1994.
3. R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *JACM*, 43(1):116–146, 1996.
4. R. Alur and T.A. Henzinger. Logics and models of real time: a survey. In *Real Time: Theory in Practice*, LNCS 600, pages 74–106. Springer, 1992.
5. E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *JACM*, 49(2):172–206, 2002.
6. G. Behrmann, A. Cougnard, A. David, E. Fleury, K. Larsen, and D. Lime. UPPAAL-Tiga: Time for playing games! In *CAV*, LNCS 4590. Springer, 2007.
7. M. Bozga, S. Graf, and L. Mounier. IF-2.0: A validation environment for component-based real-time systems. In *CAV*, LNCS 2404. Springer, 2002.
8. K. Havelund and G. Rosu. Efficient monitoring of safety properties. *STTT*, 6(2):18–173, 2004.
9. T.A. Henzinger. It's about time. In *Concur*, LNCS 1466, pages 439–454. Springer, 1998.
10. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *IC*, 111:193–244, 1994.
11. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.
12. K. G. Larsen, P. Petterson, and W. Yi. UPPAAL: Status & developments. In *CAV*, LNCS 1254, pages 456–459. Springer, 1997.
13. O. Maler, D. Nickovic, and A. Pnueli. Real time temporal logic: Past, present, future. In *FORMATS*, LNCS 3829, pages 2–16. springer, 2005.
14. O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *FORMATS*, LNCS 4202, pages 274–289. Springer, 2006.
15. O. Maler, D. Nickovic, and A. Pnueli. On synthesizing controllers from bounded-response properties. In *CAV*, LNCS 4590, pages 95–107. Springer, 2007.
16. D. Monniaux. A quantifier elimination algorithm for linear real arithmetic. In *LPAR*, LNCS 5330, pages 243–257. Springer, 2008.
17. J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS*, pages 188–197, 2005.
18. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *LMCS*, 3(3):5, 2007.
19. M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of R&D*, 3:115–125, 1959.
20. S. Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327, 1988.
21. S. Yovine. Kronos: A verification tool for real-time systems. *STTT*, 1(1–2):123–133, 1997.

A Quantifier Elimination over the Reals

We give a short introduction to quantifier elimination over the reals. We concentrate on linear inequalities that arise in the constraints defined in the paper.

Given a set of free Real variables $X = \{x_1, \ldots, x_n\}$ and free Boolean variables $B = \{b_1, \ldots, b_n\}$ an atomic formula is either b for $b \in B$, $x \bowtie y + d$, or $x \bowtie d$ for $x, y \in X$ and d an integer. The syntax of first order formulas is defined by the grammar

$$f := a \mid \neg f \mid f_1 \vee f_2 \mid \exists x.f \mid \forall x.f$$

where a is an atomic formula, and $x \in X$ is a Real variable. A formula is *quantifier free* if the last two options are not used.

Theorem 3. *Every first order formula can be converted to an equivalent quantifier free formula.*

Proof. We show how to convert a formula of the form $\exists t.f$ over the variables $\{t, x_1, \ldots, x_n\}$ to an equivalent formula over the variables $\{x_1, \ldots, x_n\}$. Multiple quantifiers and universal quantifiers are handled by repeated application of this procedure and using the equality $\forall t.f \equiv \neg\exists t.\neg f$.

Consider a formula f. We treat atomic formulas as Booleans and convert f to disjunctive normal form. That is, there is a formula $d = \bigvee_{i=1}^{k} c_i$ where c_i is a conjunction of atomic formulas. We can replace the order of existential quantification and disjunctions. Thus, $\exists t.(\bigvee_{i=1}^{k} c_i) \equiv \bigvee_{i=1}^{k}(\exists t.c_i)$. It follows that we have to be able to remove the quantification from a formula of the form $\exists t.c$ where c is a conjunction of atomic formulas. Let $c = \bigwedge_{j+1}^{m} a_j$ where a_j is an atomic formula. Order $\{a_1, \ldots, a_m\}$ as follows. Let $a_1, \ldots, a_{m_1}$ be the atomic formulas that do not contain the variable t. Let $a_{m_1+1}, \ldots, a_{m_2}$ be the atomic formulas of the form $x + d \bowtie t$, where $\bowtie \in \{<, \leq\}$, and d is an integer. Let $a_{m_2+1}, \ldots, a_m$ be the atomic formulas of the form $t \bowtie x + d$, where $\bowtie \in \{<, \leq\}$, and d is an integer.

It is simple to see that $\exists t. \bigwedge_{j=1}^{m} a_j$ is equivalent to $\bigwedge_{j=1}^{m_1} a_j \wedge \exists t.((\bigwedge_{j=m_1+1}^{m_2} a_j) \wedge (\bigwedge_{j=m_2+1}^{m} a_j))$. Let $a_j = x_j + d_j \bowtie_j t$ for $j \in \{m_1 + 1, \ldots, m_2\}$ and let $a_j = t \bowtie_j x_j + d_j$ for $j \in \{m_2 + 1, \ldots, m\}$ where $\bowtie_j \in \{<, \leq\}$ and d_j are integers forall j. Then, $\exists t.c$ is equivalent to the following formula.

$$\bigwedge_{j=1}^{m_1} a_j \wedge \bigwedge_{j=m_1+1}^{m_2} \bigwedge_{j'=m_2+1}^{m} (x_j + d_j \bowtie_j \bowtie_{j'} x_{j'} + d_{j'}),$$

where $\bowtie_j \bowtie_{j'}$ is $\leq$ if both $\bowtie_j$ and $\bowtie_{j'}$ are $\leq$ and $<$ otherwise. ⌟

We note that Theorem 3 and its proof are well known. However, as elements of the proof are required for the proof of Lemma 1 we decided to include the proof here.

The process of converting a formula to a quantifier free formula is called *quantifier elimination*. As we have to convert the formula to DNF format, the price for quantifier alternation is worst case exponential. Modern techniques for eliminating quantifiers employ SAT solvers for suggesting conjuncts that may form part of the DNF and trying to simplify them. If a conjunct simplifies to *false*, then the SAT solver learns the conflict

and adds it to the set of original constraints such that only conjuncts that do not have the same contradiction can be suggested in the future. In many cases with simple linear inequalities (as in our case) quantifier elimination works well in practice [Personal Communicatication, B. Cook]. In our case, all the free variables associated with one proposition are ordered and the formulas contain many constants, which should make quantifier elimination even simpler.

We note that quantifier elimination does not take formulas out of the subset of allowed constraints. The most complex atomic constraints allowed in our setting are comparisons of two clocks. We show that quantifier elimination cannot create an atomic constraint with a comparison of more than two clocks.

Lemma 1. *Quantifier elimination results in formulas where atomic constraints are of the form $x \bowtie d$ or $x \bowtie y + d$ for $x, y \in X$ and d and integer.*

Proof. We prove the lemma by induction on the number of eliminated quantifiers. If no quantifiers are eliminated then every atomic constraint is of the desired form. Consider a constraint that appears in a formula after one quantifier elimination. If the quantifier elimination did not affect this atomic constraint, in which case it is of the desired form. Otherwise, the constraint was obtained by quantifier elimination from two constraints of the form $x + d \leq t$ and $t \leq y + d'$. The new constraint is of the form $x \leq y + d' - d$ and maintains the required format.

B Proof of Theorem 1

Proof. The proof proceeds by induction on the structure of the formula φ. Let $f = -\mathsf{fut}(\varphi)$ and D_ψ the dependent timed automaton associated with the subformula ψ of type $\psi = \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.

$\Longrightarrow$ Assume that $\mathcal{A}_\varphi$ accepts an input word w. Then there exists an accepting run ζ of $\mathcal{A}_\varphi$ on w. Given that this run is in some state at t_1, then for every $t \in [-f, -\mathsf{fut}(\psi))$, the constraint $\psi(t)$ is satisfied at t_1 iff $(w, t_1 + t) \models \psi$.

- $\psi = p$: for the base case, if the state at time t_1 satisfies $p(t)$ with $t \in [-f, 0)$, then by definition there is a clock pair x_i^p, y_i^p s.t. $-x_i^p \leq t$ and $-y_i^p > t$. Then, at time $t_1 + t$, $-x_i^p \leq 0$ and $-y_i^p > 0$, that is the proposition monitor for p was in a state q_{2i+1}, observing p, that is $(w, t_1 + t) \models p$. Similarly, if the state at time t_1 does not satisfy $p(t)$, for all clock pairs x_i^p, y_i^p, we have $-x_i^p > t$ or $-y_i^p \leq t$, and at time $t_1 + t$, $-x_i^p > 0$ and $-y_i^p \leq 0$, that is the proposition monitor for p was in a state q_{2i}, observing $\neg p$, that is $(w, t_1 + t) \not\models p$. Hence $p(t)$ is true at t_1 iff $(w, t_1 + t) \models p$.
- $\psi = \psi_1 \mathcal{U}_{(a,b)} \psi_2$: If the state at t_1 satisfies the constraint $\psi(t) = \exists t' \in (t + a, t + b)$ s.t. $\psi_2(t')$ and $\forall t'' \in (t, t')$ $\psi_1(t'')$ with $t \in [-f, -\mathsf{fut}(\psi))$, then, by inductive hypothesis, $\exists t' \in (t_1 + t + a, t_1 + t + b)$ s.t. $(w, t') \models \psi_2$ and $\forall t'' \in (t_1 + t, t')$ $(w, t'') \models \psi_1$, and consequently, $(w, t_1 + t) \models \psi_1 \mathcal{U}_{(a,b)} \psi_2$. Conversely, if the state at t_1 does not satisfy $\psi(t)$, then $\forall t' \in (t + a, t + b)$ $\neg\psi_2(t')$ or $\exists t'' \in (t, t')$ s.t. $\psi_1(t'')$. By inductive hypothesis, $\forall t' \in (t_1 + t + a, t_1 + t + b)$ $(w, t') \not\models \psi_2$ or $\exists t'' \in (t_1 + t, t')$ s.t. $(w, t'') \not\models \psi_1$,

and consequently, $(w, t_1 + t) \not\models \psi_1 \mathcal{U}_{(a,b)} \psi_2$. Hence, $\psi(t)$ is true at t_1 iff $(w, t_1 + t) \models \psi_1 \mathcal{U}_{(a,b)} \psi_2$.

- $\psi = \psi_1 \mathcal{U}_{[b,b]} \psi_2$: the proof is similar to $\psi_1 \mathcal{U}_{(a,b)} \psi_2$.
- $\psi = \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$: We first show that D_ψ outputs p_ψ at t_1 iff $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. If D_ψ outputs p_ψ, then it is in one of the states s_1, s_2:
  - The state invariant of s_2 is

$$I_2 : \exists t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + \epsilon).\psi_2(t) \wedge \\ \forall t' \in (-\mathsf{fut}(\psi), t)\psi_1(t)$$

    By inductive hypothesis,

$$\exists t \in (t_1 - \mathsf{fut}(\psi) + a, t_1 - \mathsf{fut}(\psi) + a + \epsilon)\ .\ (w, t) \models \psi_2 \wedge \\ \forall t' \in (t_1 - \mathsf{fut}(\psi), t)\ (w, t') \models \psi_1$$

    Hence $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.
  - The state invariant of s_1 is

$$I_1 : \forall t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + a + \epsilon).\psi_1(t) \wedge \\ \forall t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + \epsilon).\neg\psi_2(t)$$

    State s_1 is unfair and the only outgoing transition goes to s_2. It follows that exists $\delta > 0$ s.t. D_ψ takes the transition at $t_1 + \delta$ where g_2 holds, and I_1 holds during $(t_1, t_1 + \delta)$. It is simple to see that g_2 corresponds to

$$\exists t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + 2\epsilon).\psi_2(t) \wedge \\ \forall t' \in (-\mathsf{fut}(\psi), t).\psi_1(t')$$

    By inductive hypothesis,

$$\forall t \in (t_1 - \mathsf{fut}(\psi), t_1 - \mathsf{fut}(\psi) + a + \epsilon + \delta)(w, t) \models \psi_1$$

    and

$$\exists t \in (t_1 - \mathsf{fut}(\psi) + a + \delta, t_1 - \mathsf{fut}(\psi) + a + 2\epsilon + \delta)(w, t) \models \psi_2(t) \wedge \\ \forall t' \in (t_1 - \mathsf{fut}(\psi) + \delta, t).(w, t') \models \psi_1$$

    Combining the two, $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.
  - If at t_1, D_ψ takes the incoming transition to s_2, the guard g_2 is satisfied. It is simple to see that g_2 corresponds to

$$\exists t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + 2\epsilon).\psi_2(t) \wedge \\ \forall t' \in (-\mathsf{fut}(\psi), t).\psi_1(t')$$

    By inductive hypothesis,

$$\exists t \in (t_1 - \mathsf{fut}(\psi) + a, t_1 - \mathsf{fut}(\psi) + a + 2\epsilon).(w, t) \models \psi_2 \wedge \\ \forall t' \in (t_1 - \mathsf{fut}(\psi), t)\ (w, t') \models \psi_1$$

    that is, $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.

* If at t_1, D_ψ takes the incoming transition to s_1, the guard g_1 is satisfied. It implies that

$$\exists t \in [-\mathsf{fut}(\psi) + a + \epsilon, -\mathsf{fut}(\psi) + a + 2\epsilon).\\ \forall t' \in (-\mathsf{fut}(\psi), t)\ \psi_1(t')$$

  By inductive hypothesis

$$\exists t \in [t_1 - \mathsf{fut}(\psi) + a + \epsilon, t_1 - \mathsf{fut}(\psi) + a + 2\epsilon).\\ \forall t' \in (t_1 - \mathsf{fut}(\psi), t)\ (w, t') \models \psi_1$$

  Let $\delta \in (0, \epsilon)$ s.t. D_ψ is in s_1 at $t_1 + \delta$. Such δ exists, because D_ψ must stay in s_1 for some strictly positive time. Then, for every $\delta' \in (0, \delta)$, we have shown that that $(w, t_1 - \mathsf{fut}(\psi) + \delta') \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. Combining the two, it follows that $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.

If D_ψ outputs $\neg p_\psi$, then D_ψ is in one of the states s_3, s_4.

* The state invariant of s_4 is

$$I_4 : \exists t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + a + \epsilon).\neg\psi(t) \wedge\\ \forall t' \in (-\mathsf{fut}(\psi) + a, t) \neg\psi_2(t')$$

  By inductive hypothesis,

$$\exists t \in (t_1 - \mathsf{fut}(\psi), t_1 - \mathsf{fut}(\psi) + a + \epsilon).(w, t) \not\models \psi_1 \wedge\\ \forall t' \in (t_1 - \mathsf{fut}(\psi) + a, t)\ (w, t') \not\models \psi_2$$

  that is $(w, t_1 - \mathsf{fut}(\psi)) \not\models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.

* The state invariant of s_3 is

$$I_3 : \forall t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + a + \epsilon).\psi_1(t) \wedge\\ \forall t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + \epsilon).\neg\psi_2(t)$$

  State s_3 is fair and the only outgoing transition goes to s_4. It follows that either
  - D_ψ stays forever in s_3
  - exists $\delta > 0$ s.t. D_ψ takes the transition at $t_1 + \delta$ where g_4 holds, and I_3 holds during $(t_1, t_1 + \delta)$.

  In the first case, ψ_2 never becomes true, so $(w, t_1 - \mathsf{fut}(\psi)) \not\models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. We now consider the second case. It is simple to see that g_4 corresponds to

$$\exists t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + 2\epsilon).\neg\psi_1(t) \wedge\\ \forall t' \in (-\mathsf{fut}(\psi), t).\neg\psi_2(t')$$

  By inductive hypothesis,

$$\forall t \in (t_1 - \mathsf{fut}(\psi), t_1 - \mathsf{fut}(\psi) + a + \epsilon + \delta)(w, t) \models \psi_1$$

  and

$$\exists t \in (t_1 - \mathsf{fut}(\psi) + a + \delta, t_1 - \mathsf{fut}(\psi) + a + 2\epsilon + \delta)(w, t) \not\models \psi_1 \wedge\\ \forall t' \in (t_1 - \mathsf{fut}(\psi) + \delta, t).(w, t') \not\models \psi_2$$

  Combining the two, $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.

* If at time t_1 D_ψ takes an incoming transition to s_4, the constraint g_4 is satisfied. It is simple to see that g_4 corresponds to

$$\exists t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + a + 2\epsilon).\neg\psi_1(t) \wedge \\ \forall t' \in (-\mathsf{fut}(\psi) + a, t)\ \neg\psi_2(t')$$

  By inductive hypothesis,

$$\exists t \in (t_1 - \mathsf{fut}(\psi), t_1 - \mathsf{fut}(\psi) + a + 2\epsilon).(w,t) \not\models \psi_1 \wedge \\ \forall t' \in (t_1 - \mathsf{fut}(\psi) + a, t)\ (w,t') \not\models \psi_2(t')$$

  that is $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.
* If at t_1, D_ψ takes the incoming transition to s_3, the guard g_3 is satisfied. It implies that

$$\exists t \in [-\mathsf{fut}(\psi) + a + \epsilon, -\mathsf{fut}(\psi) + a + 2\epsilon). \\ \forall t' \in (-\mathsf{fut}(\psi), t)\ \psi_1(t')$$

  By inductive hypothesis

$$\exists t \in [t_1 - \mathsf{fut}(\psi) + a + \epsilon, t_1 - \mathsf{fut}(\psi) + a + 2\epsilon). \\ \forall t' \in (t_1 - \mathsf{fut}(\psi), t)\ (w,t') \models \psi_1$$

  Let $\delta \in (0, \epsilon)$ s.t. D_ψ is in s_1 at $t_1 + \delta$. Such δ exists, because D_ψ must stay in s_3 for some strictly positive time. Then, for every $\delta' \in (0, \delta)$, we have shown that that $(w, t_1 - \mathsf{fut}(\psi) + \delta') \not\models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. Combining the two, it follows that $(w, t_1 - \mathsf{fut}(\psi)) \not\models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.

This finishes the proof that D_ψ outputs p_ψ at t_1 iff $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. Now, we have to consider $t \in [-f, -\mathsf{fut}(\psi))$ and there are two cases:

* If the constraint $p_\psi \wedge \forall t' \in (t, -\mathsf{fut}(\psi)].\psi_1(t)$ is satisfied, then $(w, t - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$ and $\forall t' \in (t, -\mathsf{fut}(\psi)].\psi_1(t)$ is added. By inductive hypothesis $\forall t' \in (t_1 + t, t_1 - \mathsf{fut}(\psi)]$, $(w,t') \models \psi_1$ and $(w, t_1 - \mathsf{fut}(\psi)) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. Combining the two, we conclude that $(w, t_1 + t) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. Similarly if $\neg p_\psi \vee \exists t' \in (t, -\mathsf{fut}(\psi)].\neg\psi_1(t)$, then by inductive hypothesis, $\exists t' \in (t_1 + t, t_1 - \mathsf{fut}(\psi)]$, st $(w,t') \not\models \psi_1$ or $(w, t_1 - \mathsf{fut}(\psi)) \not\models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$, that is $(w, t_1 + t) \not\models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.
* If the constraint $\exists t' \in (t + a, -\mathsf{fut}(\psi) + a].\psi_2(t') \wedge \forall t'' \in (t, t').\psi_1(t)$ is satisfied, by inductive hypothesis, $\exists t' \in (t_1 + t + a, t_1 - \mathsf{fut}(\psi) + a]$ st $(w,t') \models \psi_1$ and $\forall t'' \in (t_1 + t, t')$ $(w,t'') \models \psi_2$, hence $(w, t_1 + t) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. Conversely if the constraint is not satisfied, then $\forall t' \in (t + a, -\mathsf{fut}(\psi) + a].\neg\psi_2(t') \vee \exists t'' \in (t, t').\neg\psi_1(t)$, and by inductive hypothesis $\forall t' \in (t_1 + t + a, t_1 - \mathsf{fut}(\psi) + a]$ $(w,t') \not\models \psi_1$ and $\exists t'' \in (t_1 + t, t')$ st $(w,t'') \not\models \psi_2$, hence $(w, t_1 + t) \not\models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.

Combining all above points, we conclude that for any $t \in [-f, -\mathsf{fut}(\psi)]$, $\psi(t)$ is satisfied at t_1 iff $(w, t_1 + t) \models \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.

This completes the proof that the existence of an accepting run implies the input word satisfies the formula.

$\Longleftarrow$ Assume that for some word w we have $w \models \varphi$. We then construct an accepting run ζ of $\mathcal{A}_\varphi$ on w. For this run, it holds that if the run is in t_1, then for any subformula ψ of φ, the constraint $\psi(t)$ is true at t_1 iff $(w, t_1 + t) \models \psi$ for $t \in [-f, -\mathsf{fut}(\psi))$. We construct the run by induction on the structure of the formulas. That is, we supply the run of a dependent timed automaton for a subformula $\psi_1 \mathcal{U}_{(a,\infty)} \psi_2$ after we construct (by induction) the automata for ψ_1 and ψ_2 and establish the correctness of the constraints for ψ_1 and ψ_2. Let φ be the formula and let $f = \mathsf{fut}(\varphi)$.

- $\psi = p$: for the base case, the proposition monitor is deterministic and does not affect the acceptance of the run.
  The proof that for the (unique) run of a proposition monitor, for every time t_1 and for every $t \in [-f, 0)$ we have $(w, t_1 + t) \models p$ iff $p(t)$ holds at time t_1 is similar to the proof in the direction $\Longrightarrow$ above.
- Consider a subformula $\psi = \psi_1 \mathcal{U}_I \psi_2$, where $I = (a, b)$ or $I = [b, b]$. Notice that this formula does not add states to the automaton. By induction the runs constructed for the automata for ψ_1 and ψ_2 satisfy that $\psi_i(t)$ holds at t_1 iff $(w, t_1 + t) \models \psi_i$.
  The proof that $(w, t_1 + t) \models \psi$ iff at time t_1 the run satisfies the constraint $\psi(t)$ is similar to the proof in the direction $\Longrightarrow$ above.
- Consider a subformula $\psi = \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$. We have to show how to resolve the nondeterminism of the dependent timed automaton D_ψ constructed for ψ.
  Consider the following three conditions relating to a time t_1.
  1. There exists $t' \in (t_1 - \mathsf{fut}(\psi) + a, t_1 - \mathsf{fut}(\psi) + a + \epsilon)$ such that ψ_2 holds at time t' and for every $t'' \in (t_1 - \mathsf{fut}(\psi), t')$ we have ψ_1 holds at t''.
  2. There exists $t' \in (t_1 - \mathsf{fut}(\psi), t_1 - \mathsf{fut}(\psi) + a + \epsilon)$ such that ψ_1 does not hold at t' and for every $t'' \in (t_1 - \mathsf{fut}(\psi) + a, t')$ ψ_2 does not hold at t''.
  3. For every $t' \in (t_1 - \mathsf{fut}(\psi) + a, t_1 - \mathsf{fut}(\psi) + a + \epsilon)$ we have ψ_2 does not hold at t' and for every $t'' \in (t_1 - \mathsf{fut}(\psi), t_1 - \mathsf{fut}(\psi) + a + \epsilon)$ we have ψ_1 holds at t''.

  It is simple to see that these three conditions are mutually exclusive and their disjunction is valid.
  We now partition the time line (greater than $\mathsf{fut}(\varphi)$) into a (minimal) sequence of adjacent intervals $T_0 T_1 \ldots$ s.t. $T_i = (t_i, t_{i+1})$ with $t_{i+1} > t_i$ and $t_0 = \mathsf{fut}(\varphi)$ s.t. in every interval the input word satisfies one of the three above conditions. Note that such a partition is well-behaving (no Zeno-behavior) due to the bounded-variability of the input word.
  We associate to every interval T_i one of the states of D_ψ:
  - If T_i satisfies the condition (1), the associated state is s_2.
  - If T_i satisfies the condition (2), the associated state is s_4.
  - If T_i satisfies the condition (3), the associated state is either s_1 or s_3, resulting in the non-deterministic choice. This non-determinism is resolved by the following mutually exclusive conditions:
    - If T_{i+1} satisfies the condition (1), the state associated to T_i is s_1.
    - If T_{i+1} satisfies the condition (2), the state associated to T_i is s_3.
    - If T_i is the last interval, that is $T_i = (t_i, \infty)$, its associated state is s_3.

  The run of D_ψ is generated as a sequence of states according to the above partition of the time line w.r.t. the input word. The generated run is valid because

at any $t \in I_i$, the run is in a state whose invariant is equal to the condition of T_i, and at any time t_{i+1} between two adjacent intervals T_iT_{i+1}, there is a transition between the states in D_ψ that are associated to T_i and T_{i+1}, that is:

* Consider a transition entering state s_1. By definition of the intervals above, the only possible transitions are from s_0, s_2, and s_4. This matches the structure of the automaton.
  The guard g_1 is equivalent to $I_1 \wedge c_1$ where c_1 is

$$\exists t \in [-\mathsf{fut}(\psi) + a + \epsilon, -\mathsf{fut}(\psi) + a + 2\epsilon).$$
$$\forall t' \in [-\mathsf{fut}(\psi) + a + \epsilon, t).\psi_1(t') \wedge \neg\psi_2(t')$$

  The condition c_1 means that I_1 holds in some non-singular prefix of T_{i+1}. As I_1 holds throughout T_{i+1}, it is simple to see that I_1 holds at t_{i+1}.
  If the transition enters s_1 from s_0 then clearly $z = f$ holds as well.
* Transitions entering s_3 are similar to transitions entering s_1.
* Consider a transition entering state s_2.
  By definition of the intervals above, the only possible transitions are from s_0, s_1, and s_4. This matches the structure of the automaton.
  The guard g_2 is equivalent to $I_2 \vee (I_1 \wedge c_2)$ where c_2 is

$$\exists t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + \epsilon).\forall t' \in (-\mathsf{fut}(\psi), t).$$
$$\exists t'' \in (t' + a, t' + a + \epsilon).\psi_2(t'') \wedge \forall t''' \in (t', t'')\psi_1(t''')$$

  The condition c_2 means that I_2 holds in some non-singular prefix of T_{i+1}. As I_2 holds throughout T_{i+1}, it is simple to see that either I_2 holds at t_{i+1} or I_1 holds at t_{i+1}, depending on whether the region where ψ_2 is true is left closed or left open.
  If the transition enters s_2 from s_0 then clearly $z = f$ holds as well.
* Consider a transition entering state s_4. By definition of the intervals above, the only possible transitions are from s_0, s_2, and s_3. This matches the structure of the automaton.
  The guard g_4 is equivalent to $I_4 \vee (I_1 \wedge c_4)$, where c_4 is:

$$\exists t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + \epsilon).\forall t' \in (-\mathsf{fut}(\psi), t).$$
$$\exists t'' \in (t', t' + a + \epsilon).\neg\psi_1(t'') \wedge \forall t''' \in (t' + a, t'')\neg\psi_2(t''')$$

  The condition c_4 means that I_4 holds in some non-singular prefix of T_{i+1}. As I_4 holds throughout T_{i+1}, it is simple to see that either I_2 holds at t_{i+1} or I_1 holds at t_{i+1}, depending on whether the region where ψ_1 is false is left closed or left open.
  If the transition enters s_4 from s_0 then clearly $z = f$ holds as well.

Moreover, the generated run is accepting, since the only unfair state in D_ψ is s_1, and the run can be in that state at some $t \in I_i$ only if I_i is followed by I_{i+1} that satisfies condition (1), that is by state s_2.
The proof that for every t_1 we have D_ψ outputs p_ψ at t_1 iff $(w, t_1 - \mathsf{fut}(\psi)) \models \psi$ is similar to the proof in the direction $\Longrightarrow$ above. Based on the correctness of p_ψ, the proof that at time t_1 the constraint $\psi(t)$ holds iff $(w, t_1 + t) \models \psi$ is also similar to the proof in the direction $\Longrightarrow$ above.

This completes the proof that for every formula φ, if the input word satisfies the formulae, there is an accepting run of $\mathcal{A}_\varphi$ induced by that word.

⌟