From MTL to Deterministic Timed Automata*

Dejan Ničković[1] and Nir Piterman[2]

[1] IST, Klosterneuburg, Austria
[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 include all future operators. 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 determinize our automata in an exponential construction that does not increase the number of clocks. This leads to a doubly exponential construction from MTL to deterministic timed automata, compared with triply exponential using existing approaches.
We offer an alternative to the existing approach to linear real-time model checking, which has never been implemented. It further offers a unified framework for model checking, runtime monitoring, and synthesis, in an approach that can reuse tools, implementations, and insights from the discrete setting.

1 Introduction

The ability to write high-level, intuitive specifications in temporal logic is what, in many cases, underlies the success of applications in model checking, runtime monitoring, and controller synthesis. In this paper we concentrate on applications where linear-time temporal logic (LTL) [20] is used to describe ongoing behaviors. In the case of model-checking the specification φ is effectively translated to a nondeterministic automaton $\mathcal{A}_\varphi$ that is composed with the system and analyzed in the search of *bad* behaviors. In runtime monitoring, it is not enough to translate φ to a nondeterministic automaton. Indeed, an individual behavior of the system may induce multiple runs, which have to be followed simultaneously. Resolving nondeterminism on-the-fly induces an extra computational cost at every step of the monitoring. Thus, it is best to use a *deterministic* automaton. As monitoring is usually concerned only with finite input sequences, a simple determinization, based on the subset construction, suffices. Finally, in controller synthesis, we would like to automatically generate an implementation of a system S that satisfies a specification φ [24]. Synthesis requires to convert φ to a deterministic automaton, however, full determinization of ω-automata is required [26, 23].

Here, we are interested in these three applications in the context of real-time. The main model for reasoning quantitatively about time is timed automata [3]. Timed automata are suitable for modeling certain time-dependent phenomena, and their reacha-

* Part of this work was done while both authors were at Verimag, France. The first author is supported in part by the EU project COMBEST and the EU Network of Excellence ARTIST-DESIGN. The second author is supported by the grant UK EPSRC EP/E028985/1.

bility (or empty language) problem is decidable, facts that have been exploited in verification tools, e.g., Kronos [29] and Uppaal [15].

As in the untimed case, we would like to combine the model of timed automata with a powerful logic and apply algorithms for model checking, runtime monitoring, and controller synthesis. Many variants of real-time logics [14, 5, 12, 11] have been proposed. However, unlike in 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 [4] as a restriction of the logic MTL [14]. 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 [4] by converting an MITL formula to a nondeterministic timed automaton and analyzing the structure of that automaton. Further investigations of MITL and MTL suggested alternative translations of MITL to nondeterministic timed automata [17, 18] and used alternating timed automata to show decidability of MTL in certain circumstances [22].

The mentioned translations of MITL to nondeterministic timed automata provide the necessary theory for MITL model checking. However, to the best of our knowledge, (due to their complexity) there are no tools implementing linear-time model checking of timed systems. In addition, these constructions do not offer a viable solution for runtime monitoring or controller synthesis, as they do not produce *deterministic* automata. In general, for MITL it is impossible to construct deterministic automata [16]. 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. A deterministic automaton for φ would require infinite memory to remember all possible occurrences of p within $(0, a)$. Even if we find a fragment of MITL that can be translated to deterministic timed automata, the known constructions [4, 17] cannot be used, as arbitrary timed automata cannot be determinized [3]. Determinization constructions rely on the ability to create a finite representation of a set of runs. When clocks are involved, it is impossible to finitely represent all the values of the clocks in a deterministic timed automaton. Even worse, the realizability problem for MITL is undecidable [10]; and for MTL, realizability is undecidable even for finite words, but becomes decidable if one restricts the number of clocks used by the controller [8].

Synthesis from specifications given as deterministic timed automata is possible [21]. In order to produce deterministic automata one has to resolve two sources of nondeterminism in timed automata arising from MITL:

1. *Unbounded variability:* there can be an unbounded number of events in a fixed interval of time, which the automaton needs to remember. It follows that one cannot fix a bound on the number of clocks that the timed automaton needs to use for memorizing relevant events. Most examples showing that timed automata cannot be determinized or complemented use unbounded variability.
2. *Acausality:* the satisfaction of a formula at time t may depend on values of inputs at time $t' > t$. In order to determine whether the input sequence satisfies the formula at time t, the automaton "predicts" the future values of inputs, and aborts later

(at t') computations corresponding to wrong predictions. Acausality is a source of nondeterminism for both untimed and real-time temporal logics.

Wilke showed that removing the first source of nondeterminism (i.e., bounding the variability of input signals) enables complementation and determinization of timed automata [28]. His proof uses monadic-second order logic and offers no practical solution. Several classes of timed automata that can be determinized are identified in [7]. These automata include strongly non-Zeno timed automata, which reject words that vary more than a certain bound. The construction in [7] is impractical for MITL specifications as it is doubly exponential and uses an exponential number of clocks; leading to a triple exponential construction with a doubly exponential number of clocks. Also, determinization is considered only for finite words.

In [18] a construction from MTL under bounded variability to deterministic timed automata is suggested. This solution, however, only partially solves acausality: only invariants "always φ" are handled, where φ is an MTL formula with past temporal operators and *bounded* future operators. Bounded future operators in the scope of an always are syntactically changed to past operators. As the past is naturally deterministic [16], the standard construction of [17] produces a deterministic automaton. Compared with [7], the construction is single exponential instead of triple exponential.

It follows that bounded variability is a practical solution for constructing deterministic timed automata from real-time specifications. For many applications, bounded variability is a very reasonable assumption [6]. High variability arises from the frequency at which systems operate; and almost all systems have a bound on the frequencies in which they operate. In general, high frequencies are hard to produce making systems more complicated. In practice, bounded variability covers the most interesting cases.

Using the assumption of bounded-variability, we show how to fully handle acausality of MTL semantics. For that, we devise a completely new construction for converting MTL formulas to nondeterministic automata, which relies on bounded variability. Our construction separates the timed automata into two parts: (i) a standard timed automaton that monitors changes in the inputs and memorizes events with clocks; and (ii) a *dependent timed automaton*, that makes passive use of clocks controlled by the first part. The first part is *deterministic* by construction. The second part, namely the dependend timed automaton, generates *discrete* predictions regarding future events.

We then show that dependent timed automata with the following properties can be determinized: (i) transitions cannot be enabled throughout the stay in a state (no dense nondeterminism); and (ii) when a transition is enabled the automaton can stay for a little while in the target state. Both conditions hold for the automata we construct. The determinization itself is a slight variant of determinization for untimed finite automata on infinite words [26, 23]. The determinization is exponential and does not change the number of clocks. The overall result is doubly exponential construction from MTL to deterministic timed automata.[3]

Our new construction has many additional advantages even when producing nondeterministic automata. In our construction the number of clocks depends on the number of propositions, the depth of the longest chain of *nested* timed operators, and the

[3] In contrast, an exponential determinization for general timed automata under bounded variability is impossible. The doubly exponential determinization in [7] is optimal.

bounded variability of the input. Updates to clocks are extremely simple and depend directly on the input. In previous constructions clocks are allocated according to variability and depth of each operator separately. In our construction the number of states associated with every unbounded-until is constant, while in previous constructions *every* temporal operator requires states that are proportional to the number of active intervals that the temporal operator may have. For the fragment of bounded future operators considered in [18] our automata are deterministic by construction and do not require the extra determinization step. Finally, previous translations read "singular" (zero-duration) and "open" intervals, while in our construction automata use only left-closed right-open intervals, which matches existing tools, such as Kronos [29] and Uppaal [15].

2 Signals and MTL

Let AP be a set of atomic propositions. A signal over AP is a function $w : \mathbb{T} \to 2^{AP}$, where $\mathbb{T}$ is either the non-negative reals $\mathbb{R}_{\geq 0}$ (infinite-length) or an interval $[0, r)$ (finite length). 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 $t \geq r_1$. Consider signal w over AP and signal w' over AP' of the same length such that $AP \cap AP' = \emptyset$. Their product $w \times w'$ is the signal such that for $p \in AP$ we have $(w \times w')_p = w_p$ and for $p \in AP'$ we have $(w \times w')_p = w'_p$. Product is defined for sets of words in the natural way. We say that w is of *bounded-variability* l if for every proposition p, every $t \in \mathbb{T}$, and every $t_0 < t_1 < \cdots < t_l \in [t, t + 1)$ there is i such that $w_p[t_i] = w_p[t_{i+1}]$. That is, proposition p changes its value at most $l - 1$ times in every interval of length 1. In this paper we consider only signals of bounded variability.

The syntax of the future fragment of MTL interpreted over dense-time signals 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 $AP = \{p_1, \ldots, p_n\}$ of propositions and I is an interval of one of the following forms: $[b, b],\ (a, b),$ or (a, ∞) where $0 \leq a < b$ are integers. An until $\mathcal{U}_I$ is *unbounded* if I is (a, ∞), otherwise it is *bounded*.

The semantics of an MTL formula φ with respect to a signal w is defined recursively via the satisfiability relation $(w, t) \models \varphi$, indicating that signal w satisfies φ at time t:

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

A formula φ is satisfied by w if $(w, 0) \models \varphi$ and $L(\varphi) = \{w \mid (w, 0) \models \varphi\}$. Other Boolean and temporal operators are derived as usual. In particular, $\Diamond_I \varphi = \mathrm{T}\ \mathcal{U}_I\ \varphi$ and $\Box_I \varphi = \neg \Diamond_I \neg\varphi$ are the *eventually* and *always* operators. Similarly, intervals such as $[a, b]$, $[a, b)$, $[a, \infty)$ can be expressed. When $I = (0, \infty)$, we simply omit it.

We suggest a new construction from MTL under bounded variability to timed automata. The construction is based on computing a bound f such that the automaton at time t memorizes the input signal at the interval $[t - f, t)$. The truth value of the formula is computed with a delay: when the automaton is reading time t it computes the value

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

Fig. 1. The function $\mathsf{fut}()$

of the formula for time $t - f$. The function fut, which determines the interval's size, is defined in Figure 1.

We compute the truth value of a formula φ at time t by looking on the interval $[t, t + \mathsf{fut}(\varphi))$. The only non-trivial part of this definition is the unbounded until $\varphi = \varphi_1 \mathcal{U}_{(a,\infty)} \varphi_2$, which, apart from a, requires two additional lookaheads ($a + 2$). In this paragraph, we give an informal overview of the ingredients needed for our construction of determinizable timed automata from an unbounded until formula. The formal definition of the construction is presented in Section 4 and illustrated by Figure 4. We want an automaton to know for sure that φ is true without remembering how long φ_1 held. For that, the interval $(t, t + a]$ is *never* sufficient: if φ_1 holds throughout $(t, t + a]$, we must guess that φ_1 continues until φ_2 holds later and if φ_1 holds until φ_2 holds *within* $[t, t + a)$, then we need to know whether φ_1 held also before t. Thus, we memorize φ_1 and φ_2 in $(t, t + a + 1]$, which is *sometimes* sufficient to know for sure that φ is true. For example, φ holds for sure if φ_1 holds throughout $(t, t + a + 0.5]$ and φ_2 holds at $t + a + 0.5$. We use the information recorded in $(t, t + a + 1]$, combined with a purely **discrete** guess about the satisfaction of the *untimed* formula $\varphi_1 \mathcal{U}_{\geq 0} \varphi_2$ at time $t + a + 1$, to construct an automaton that determines the truth value of φ at t. It distinguishes between three situations: (1) φ_2 is true at some $t' \in (t + a, t + a + 1]$ and φ_1 holds throughout (t, t'), hence φ clearly holds at t; (2) φ_1 does not hold at some $t' \in (t, t + a + 1]$ and φ_2 is false during $(t + a, t')$, hence φ is false at t; (3) φ_1 is true during $(t, t + a + 1]$ and φ_2 is false throughout $(t + a, t + a + 1]$, and we need to predict the truth value of $\varphi_1 \mathcal{U}_{\geq 0} \varphi_2$ at $t + a + 1$. The automaton is nondeterministic: it makes *wrong* guesses and aborts runs. We use the additional lookahead to eliminate one type of wrong guess. Consider, for example, some $t > 0$ and the case that φ_1 holds continuously throughout $[0, \infty)$ and φ_2 holds in $(0, t + a)$ and $(t + a + 1, \infty)$. For every $t' < t$ we have φ holds at time t' and our automaton *sees* that φ is true. At exactly time t, the automaton is blind: φ_2 does not hold throughout $(t + a, t + a + 1]$. It has to guess whether φ continues to hold. If it guesses that φ is false, then at every $t'' > t$ it realizes its mistake and aborts as, indeed, φ_2 holds at $t'' + a + 1$. Hence, the "length of error" was 0: exactly at time t. We eliminate 0-duration errors using the second lookahead $t + a + 2$, which provides enough 'prediction' power to avoid such errors. It follows that the 0-duration error elimination results in an automaton that remains non-deterministic, but this step is important for its determinization described in Section 5.

The function fut is used also to decide where the truth value of a formula expires. For example, if at time t we know the value of q and r in $[t - 8, t)$, our knowledge for the formula $q \mathcal{U}_{(2,\infty)} r$ expires at time $t - 2 - 2$. As $q \mathcal{U}_{(2,\infty)} r$ is unbounded, this truth value may depend further on a guess.

3 Timed Automata

We use a variant of timed automata that differs slightly from the classical definitions [3, 13, 1]. Our automata read multi-dimensional *dense-time* Boolean signals and output Boolean signals. Input and output are associated with states *and* 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 $A \subseteq \mathbb{R}^n_{\geq 0}$ we use $cl(A)$ to denote its (topological) closure.

The set of valuations of a set $\mathcal{C} = \{x_1, \ldots, x_n\}$ 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 view a constraint $c \in \mathbb{C}(\mathcal{C})$ as a subset $c \subseteq \mathcal{H}$. We also 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. In the full version we show that application of quantifier elimination results in clock constraints in $\mathbb{C}(\mathcal{C})$. We used the tool in [19] to eliminate quantifiers in some of the examples below.

Definition 1. *A* timed automaton (TA) *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 the guard* True *and use only updates of the form $x := 0$. We consider generalized Büchi automata, where $\mathcal{F} \subseteq 2^Q$, and parity automata, where $\mathcal{F} = \langle F_0, \ldots, F_k \rangle$ partitions Q. Unless mentioned explicitly, automata are of the first type.*

The behavior of a TA as it reads a signal w consists of a strict alternation between time progress periods, where the TA 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 TA is one of the following:

- A time step $(q, v) \xrightarrow{\sigma^t} (q, v + t)$ $t \in \mathbb{R}_{>0}$ where $\sigma = \lambda(q)$ 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 TA 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 $\sum_i t_i$ diverges. We denote by $\zeta[t]$ the valuation of the clocks in ζ at time t. That is, at $t = \sum_{j=1}^{i} t_j$ we have $\zeta[t] = v_{i+1}$ and for $t' < t_{i+1}$ we have $\zeta[t+t'] = \zeta[t]+t'$. Given a run ζ, the set $inf(\zeta)$ is the set of states q such that the set of time instants in which q is visited is unbounded. A run is accepting according to the generalized Büchi condition $\mathcal{F}$ if for every $F \in \mathcal{F}$ we have $F \cap inf(\zeta) \neq \emptyset$. A run is accepting according to the parity condition $\mathcal{F}$ if the minimal i such that $F_i \cap inf(\zeta) \neq \emptyset$ is even. 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 $w(t) = \sigma_i$. An input signal is *accepted* if it is carried by an accepting run. The *language* of A, denoted $L(A)$, is the set of accepted input signals.

Let $A_i = \langle \Sigma_i, Q_i, \mathcal{C}_i, \lambda_i, I_i, \Delta_i, q_0^i, \mathcal{F}_i \rangle$, for $i \in \{1, 2\}$, be two TA such that $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$. The composition $A_1 \parallel A_2 = \langle \Sigma_1 \times \Sigma_2, Q_1 \times Q_2, \mathcal{C}_1 \cup \mathcal{C}_2, \lambda, I, \Delta, (q_0^1, q_0^2), \mathcal{F} \rangle$, 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 relation Δ includes:

- *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 $g = I_1(q_1) \wedge g_2$.

These three types of transitions reflect the asynchronicity of the composed TA, allowing one to take a discrete step while the other is in the middle of a time step. Note that the alphabets of A_1 and A_2 are disjoint.

Lemma 1. $L(A_1 \parallel A_2) = L(A_1) \times L(A_2)$

A TA is *deterministic* if from every reachable configuration every event and every 'non-event' leads to exactly one 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* $(t, t') \cap g \neq \emptyset$*.*

Dependent timed automata (DTA) are transducers of runs of TA. Accordingly, DTA have output and composition of DTA allows one automaton to read the output of the other. DTA passively read clocks of other TA and have no clocks of their own. Thus, they *depend* on other TA to supply them the clocks.

Definition 3. *A* dependent timed automaton (DTA) *is* $B = \langle \Sigma, \Gamma, Q, \mathcal{C}, \gamma, I, \Delta, q_0, \mathcal{F} \rangle$*, where* Σ*,* Q*,* $\mathcal{C}$*,* q_0*, and* $\mathcal{F}$ *are as in timed automata. DTA have, in addition, an output alphabet* Γ*, and an output function* $\gamma : Q \to \Gamma$*. The labeling function is replaced by a more general staying condition* I *that assigns to every state a Boolean combination of atomic constraints 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 a set* AP *of propositions and freely use propositions to define staying conditions and transition guards.*

Let ζ be the run of TA $\mathcal{A}$ on input signal w. A *run* of the DTA reading ζ and w is a finite or infinite sequence of states and transitions, where the states are annotated by the time the DTA stayed in them. Formally, $\zeta' : q_0 \xrightarrow{\delta_0} q_1^{t_1} \xrightarrow{\delta_1} q_2^{t_2} \cdots$, such that for every $i \geq 0$, there are g_i and o_i such that $\delta_i = (q_i, g_i, o_i, q_{i+1})$ and g_i is satisfied by the values $\zeta[\vec{t_i}]$ and $w[\vec{t_i}]$, where $\vec{t_i} = \sum_{j=1}^{i} t_j$. Furthermore, for every $t \in (\vec{t_i}, \vec{t_{i+1}})$ we have $I(q_i)$ is satisfied by $\zeta[t]$ and $w[t]$. Acceptance is defined as for runs of TA. An accepting run ζ' carries the signal w' over $\Sigma \times \Gamma$ such that $w'_{\Sigma} = w$, $w'[\vec{t_i}]_{\Gamma} = o_i$, and for all $t \in (\vec{t_i}, \vec{t_{i+1}})$ we have $w'[t]_{\Gamma} = \gamma(q_i)$. Given a TA run ζ carrying signal w, we denote by $B(w, \zeta)$ the set of signals carried by accepting runs of B.

Consider two DTA $B_i = \langle \Sigma_i, \Gamma_i, Q_i, \mathcal{C}, \gamma_i, I_i, \Delta_i, q_0^i, \mathcal{F}_i \rangle$ 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 DTA. Let $B_1 \otimes B_2 = \langle \Sigma_1, \Gamma_1 \times \Gamma_2, Q_1 \times Q_2, \mathcal{C}, \gamma, I, \Delta, (q_0^1, q_0^2), \mathcal{F} \rangle$, where $\gamma(q_1, q_2) = (\gamma_1(q_1), \gamma_2(q_2))$ and $\mathcal{F} = \{S \times T \mid S \in \mathcal{F}_1 \text{ and } T = Q_2, \text{ or } S = Q_1 \text{ and } T \in \mathcal{F}_2\}$. 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 relation Δ includes:

- 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 $g = g_1 \wedge simp(o_1, g_2)$,
- 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
- 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 $g = I_1(q_1) \wedge simp(\gamma_1(q_1), g_2)$.

Lemma 2. *For every run* ζ *and signal* w*,* $B_1 \otimes B_2(w, \zeta) = B_2(B_1(w, \zeta), \zeta)$.

Consider a TA $A_1 = \langle \Sigma_1, Q_1, \mathcal{C}, \lambda_1, I_1, \Delta_1, q_0^1, \mathcal{F}_1 \rangle$ and a DTA $B_2 = \langle \Sigma_1, \Gamma_2, Q_2, \mathcal{C}, \gamma_2, I_2, \Delta_2, q_0^2, \mathcal{F}_2 \rangle$. Their composition $A_1 \otimes B_2$ is the TA $\langle \Sigma_1, Q_1 \times Q_2, \mathcal{C}, \lambda, I, \Delta, (q_0^1, q_0^2), \mathcal{F} \rangle$, where $\lambda(q_1, q_2) = \lambda_1(q_1)$, and $\mathcal{F} = \{S \times T \mid S \in \mathcal{F}_1 \text{ and } T = Q_2, \text{ or } S = Q_1 \text{ and } T \in \mathcal{F}_2\}$. The staying condition is $I(q_1, q_2) = I_1(q_1) \wedge simp(\lambda_1(q_1), I_2(q_2))$. The transition relation Δ includes:

- simultaneous transitions $((q_1, q_2), g, \rho_1, (q_1', q_2'))$, where $(q_1, g_1, \rho_1, q_1') \in \Delta_1$, $(q_2, g_2, o_2, q_2') \in \Delta_2$, $g = g_1 \wedge app(\rho_1, simp(\lambda_1(q_1'), g_2))$, and $app(\rho_1, g_2)$ applies the effect of ρ_1 on g_2, e.g., if ρ_1 includes $x := y$ we replace x in g_2 by y,
- 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 $g = g_1 \wedge app(\rho_1, simp(\lambda_1(q_1'), \gamma_2(q_2)))$, and
- 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 $g = I_1(q_1) \wedge simp(\lambda_1(q_1), g_2)$.

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

Fig. 2. The transition of proposition monitor.

Lemma 3. *$L(A_1 \otimes B_2) = \{w \mid \exists \zeta_1$ accepting run of A_1 carrying w and $B_2(w, \zeta_1) \neq \emptyset\}$.*

As TA have no output, the composition of a TA with a DTA removes the output. Thus, as in the composition $B_1 \otimes B_2$ DTA B_2 may read the output of B_1, the composition $A \otimes B_1 \otimes B_2$ should be read as $A \otimes (B_1 \otimes B_2)$. If A_1 is generalized Büchi with $\mathcal{F}_1 = \{Q_1\}$ and B_1 is parity with $\mathcal{F}_2 = \langle F_0, \ldots, F_k \rangle$ their composition $A_1 \otimes B_2$ is a parity automaton with $\mathcal{F} = \langle F'_0, \ldots, F'_k \rangle$, where $F'_i = Q_1 \times F_i$. We do not define other compositions of parity conditions.

4 MTL to Nondeterministic Timed Automata

We suggest a novel construction for the conversion of MTL formulas to nondeterministic TA. The advantage of this construction is that it effectively distinguishes between discrete guesses relating to occurrences in the future (made by DTA) and the accumulation of knowledge with clocks (made by deterministic and extremely simple TA). This separation allows us to construct a deterministic automaton for the formula in Section 5. This section starts by introducing proposition monitors, deterministic TA that log information about the input. We then show how to construct the DTA that handle general MTL formulas. Note that the number of clocks depends on the structure of the formula through the function fut and the construction of the proposition monitors.

We introduce a TA that memorizes the times in which a proposition is true. 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 bounded-variability value (i.e., the limit on 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 $n = \lceil \frac{fk}{2} \rceil$. Formally, for a proposition p, let $\mathcal{A}_p = \langle 2^{\{p\}}, Q, \mathcal{C}, \lambda, I, \Delta, q_{in}, \{Q\} \rangle$, where $Q = \{q_{in}, q_0, \ldots, q_{2n}\}$, $\lambda(q_{2i}) = \emptyset$, $\lambda(q_{2i+1}) = \{p\}$, $\mathcal{C} = \{x_1^p, \ldots, x_n^p, y_1^p, \ldots, y_n^p\}$, for $j > 1$ we have $I(q_j) = y_1^p < f$ and $I(q_0) = I(q_1) = \mathit{True}$ and Δ is given in Figure 2. One such proposition monitor is given in Figure 3.

We use the TA $\mathcal{A}_z$ with one state and one clock z, which measures the time since time 0, to check whether the bound f has been reached. Formally, $\mathcal{A}_z = \langle 2^{\emptyset}, \{q_{in}, q\},$

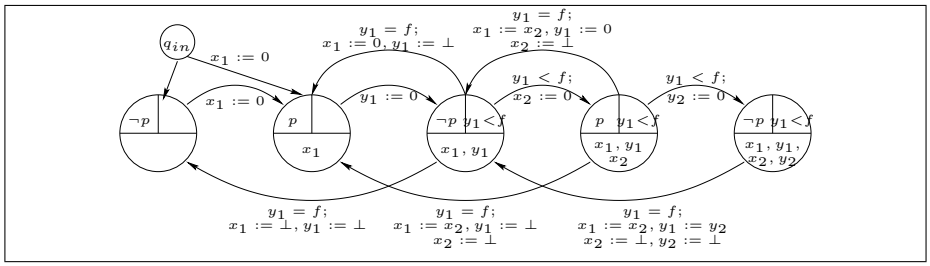

Fig. 3. Proposition monitor for p, where $f = \mathsf{fut}(\varphi)$ and $\lceil \frac{fk}{2} \rceil = 2$. State labels consist of: the label p or $\neg p$; the invariant true (blank) or $y_1 < f$; and clocks active in the state. Transition labels consist of: the transition guard, e.g., $y_1 < f$; and the clock update, e.g., $y_1 := \bot$.

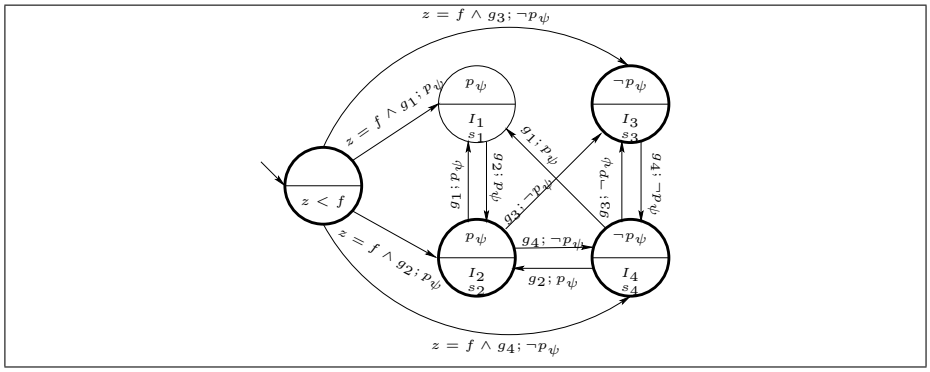

Fig. 4. A DTA for unbounded until. States s_1 and s_3 correspond to $\square_{(0,a+1)}\,\varphi_1 \wedge \neg(\varphi_1 \mathcal{U}_{(a,a+1)}\varphi_2)$, state s_2 corresponds to $\varphi_1 \mathcal{U}_{(a,a+1)}\varphi_2$, and state s_4 corresponds to $\neg(\square_{(0,a+1)}\,\varphi_1) \wedge \neg(\varphi_1 \mathcal{U}_{(a,a+1)}\varphi_2)$.

$\{z\}, \lambda, I, \Delta, q_{in}, \{q\}\rangle$, where $\lambda(q) = \mathit{True}$, $I(q) = z > 0$, and $\Delta = \{(q_{in}, \mathit{True}, \{z := 0\}, q)\}$. Let $\mathcal{A}_m$ denote the composition of all proposition monitors and $\mathcal{A}_z$.

We turn to the construction of DTA. Consider an MTL formula φ. We construct a sequence of DTA that use each other's outputs and eventually are all composed with $\mathcal{A}_m$, which supplies all the clocks that are read by them. For most subformulas ψ of φ, the truth value of ψ can be deduced from the values of clocks in $\mathcal{A}_m$ and the DTA of their subformulas. For such formulas, we define constraints $c_\psi(t)$ that depend on mentioned DTA and $\mathcal{A}_m$ and, intuitively, characterize the truth value of ψ at time t. These constraints are ultimately staying conditions and transition guards in DTA for superformulas of ψ. For an unbounded subformula ψ (i.e., Until where the upper limit is ∞) we construct a small DTA that computes the value of ψ at exactly the time point $\mathsf{fut}(\psi) - \mathsf{fut}(\varphi)$ (i.e., $\mathsf{fut}(\varphi) - \mathsf{fut}(\psi)$ before the current time point). This DTA uses constraints defined for subformulas of ψ. Similar to other formulas, based on the DTA for ψ, we also define the constraint $c_\psi(t)$, which characterizes the truth value of ψ at other times $t \neq \mathsf{fut}(\psi) - \mathsf{fut}(\varphi)$. This constraint is again used for staying conditions and transition guards in superformulas of ψ. Finally, we construct a DTA for the formula φ itself. Then, we take the composition of all DTA constructed and compose them with $\mathcal{A}_m$ to produce the TA that accepts the language of φ. Formally, we have the following.

The mentioned constraints are in the first-order theory of the reals with clocks of $\mathcal{A}_m$ as free variables. By eliminating quantifiers these constraints can be converted to clock constraints that are used in staying conditions and transition guards. 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 $[-2.37, -1.49)$. Formally, we define by induction the DTA and constraints. For a subformula ψ, the constraint $C_\psi(t)$ is valid for $t \in [-f, -\mathsf{fut}(\psi))$. We assume that quantifier elimination is applied on all constraints.

- For a proposition p and for $t \in [-f, 0)$. We define $C_p(t) = \bigvee_i . -x_i^p \leq t \wedge -y_i^p > t$. The finite disjunction on i depends on the number of clocks in $\mathcal{A}_p$ (part of $\mathcal{A}_m$).
- For subformula ψ of the form $\neg\psi_1$, $\psi_1 \vee \psi_2$, or $\psi_1 \wedge \psi_2$ we define $C_\psi(t)$ using $C_{\psi_1}(t)$ and $C_{\psi_2}(t)$ in the obvious way. The range allowed for t is the minimal range allowed by C_{ψ_1} and C_{ψ_2}. For example, we define $C_{\psi_1 \vee \psi_2}(t) = C_{\psi_1}(t) \vee C_{\psi_2}(t)$.
- For $\psi = \psi_1 \mathcal{U}_I \psi_2$, 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 $C_{\psi_1}(t)$ is defined for $t \in [-f, -\mathsf{fut}(\psi_1))$ and $C_{\psi_2}(t)$ is defined for $t \in [-f, -\mathsf{fut}(\psi_2))$. So, for $t \in [-f, -\mathsf{fut}(\psi))$ it is always the case that $t + b$ is in the range where $C_{\psi_1}(t)$ and $C_{\psi_2}(t)$ are defined. In the case that $I = (a, b)$, for $t \in [-f, -\mathsf{fut}(\psi))$ we formally define $C_\psi(t) = \exists t' \in (t + a, t + b).C_{\psi_2}(t') \wedge \forall t'' \in (t, t').C_{\psi_1}(t'')$. In the case that $I = [b, b]$, for $t \in [-f, -\mathsf{fut}(\psi))$ we formally define $C_\psi(t) = C_{\psi_2}(t + b) \wedge \forall t' \in (t, t + b).C_{\psi_1}(t')$.
- Consider a formula $\psi = \psi_1 \mathcal{U}_{(a,\infty)} \psi_2$.
  By definition[4] $\mathsf{fut}(\psi) = a + 2 + max(\mathsf{fut}(\psi_1), \mathsf{fut}(\psi_2))$. It follows that $C_{\psi_1}(t)$ is defined for $t \in [-f, -\mathsf{fut}(\psi_1))$ and $C_{\psi_2}(t)$ is defined for $t \in [-f, -\mathsf{fut}(\psi_2))$. So, for $t \in [-f, -\mathsf{fut}(\psi)]$ it is always the case that $(t, t + a + 2)$ is contained in the range where C_{ψ_1} and C_{ψ_2} are defined. We construct a DTA for ψ and use its output for defining C_ψ.
  - We construct a DTA for the truth value of ψ at time $t = -\mathsf{fut}(\psi)$. Formally, let $\mathcal{B}_\psi$ be the automaton in Figure 4, where the guards and the invariants are as follows.

$$
\begin{array}{ll}
I_1, I_3 & : \forall t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + a + 1).C_{\psi_1}(t) \wedge \\
 & \qquad \forall t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + 1).\neg C_{\psi_2}(t) \\
I_2 & : \exists t \in (-\mathsf{fut}(\psi) + a, -\mathsf{fut}(\psi) + a + 1). \quad C_{\psi_2}(t) \wedge \forall t' \in (-\mathsf{fut}(\psi), t).C_{\psi_1}(t) \\
I_4 & : \exists t(-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + a + 1).\neg C_{\psi_1}(t) \ \wedge \forall t' \in (-\mathsf{fut}(\psi) + a, t).\neg C_{\psi_2}(t) \\
g_1, g_3 & : I_1 \wedge \exists t \in [-\mathsf{fut}(\psi) + a + 1, -\mathsf{fut}(\psi) + a + 2). \\
 & \qquad \forall t' \in [-\mathsf{fut}(\psi) + a + 1, t).C_{\psi_1}(t') \wedge \neg C_{\psi_2}(t') \\
g_2 & : I_2 \vee (I_1 \wedge \exists t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + 1). \\
 & \quad \forall t' \in (-\mathsf{fut}(\psi), t).\exists t'' \in (t' + a, t' + a + 1).C_{\psi_2}(t'') \wedge \forall t''' \in (t, t'')C_{\psi_1}(t''') \\
g_4 & : I_4 \vee (I_3 \wedge \exists t \in (-\mathsf{fut}(\psi), -\mathsf{fut}(\psi) + 1).\forall t' \in (-\mathsf{fut}(\psi), t). \\
 & \qquad \exists t'' \in (t', t' + a + 1).\neg C_{\psi_1}(t'') \wedge \forall t''' \in (t' + a, t'')\neg C_{\psi_2}(t''')
\end{array}
$$

    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.
    Intuitively, the invariants realize the intuitive meaning of the states as explained in Figure 4. The transition guards, in addition, use the extra 1 in order to make sure that when going to the next state it is possible to *stay* in it (eliminate 0-duration

[4] We note that replacing $a + 2$ by $a + 2\epsilon$ for every $\epsilon > 0$ would not affect the correctness of the construction. We choose integer values for the lookaheads to avoid normalizing the largest constant in the guards and invariants of the timed automaton

errors). This is required in order to be able to apply determinization. In case that determinization is not pursued, the extra lookahead can be removed and the guards are I_1 for transitions into s_1 and s_3, $I_2 \vee I_1$ for transitions into s_2, and $I_4 \vee I_3$ for transitions into s_4.

- For every $t \in [-\mathsf{fut}(\varphi), -\mathsf{fut}(\psi)]$ the constraint $C_\psi(t)$ that describes the truth value of ψ is formally defined as follows:

$$(t = -\mathsf{fut}(\psi) \wedge p_\psi) \vee (t < -\mathsf{fut}(\psi) \wedge p_\psi \wedge \forall t' \in (t, -\mathsf{fut}(\psi)].C_{\psi_1}(t)) \vee$$
$$(t < -\mathsf{fut}(\psi) \wedge \exists t' \in (t + a, -\mathsf{fut}(\psi)).C_{\psi_2}(t') \wedge \forall t'' \in (t, t').C_{\psi_1}(t))$$

This completes the inductive part of the construction[5]. Let $\psi_1, \ldots, \psi_n$ be all the unbounded temporal operators in φ such that if ψ_i is a subformula of ψ_j then $i < j$. Let $\mathcal{B}_{\psi_1}, \ldots, \mathcal{B}_{\psi_n}$ be the DTA constructed in the inductive part, $AP' = \{p_{\psi_1}, \ldots, p_{\psi_n}\}$, and $\Gamma = 2^{AP'}$. That is, Γ includes the output of all DTA constructed by induction. Note that for every i we have $\mathcal{B}_{\psi_i}$ is merely informative. That is, $\mathcal{B}_{\psi_i}$ accepts all inputs and adorns them with the proposition p_{ψ_i}.

We now construct a final DTA $\mathcal{B}$ for φ itself. Let $\mathcal{B}$ be $\langle \Sigma \times \Gamma, \{o\}, \{q_0, q_1, q_2\}, \mathcal{C}, \gamma, I, \Delta, q_0, \{\{q_1\}\}\rangle$, 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 $g_1 = C_\varphi(-f) \wedge z = f$ and $g_2 = \neg C_\varphi(-f) \wedge z = f$. That is, $\mathcal{B}$ 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. Let $\mathcal{B}_\varphi = \mathcal{B}_{\psi_1} \otimes \cdots \otimes \mathcal{B}_{\psi_n} \otimes \mathcal{B}$. Finally, the TA for φ, denoted by A_φ, is $\mathcal{A}_m \otimes \mathcal{B}_\varphi$.

Lemma 4. $L(\mathcal{A}_\varphi) = L(\varphi)$

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 $2m\lceil \frac{k \cdot \mathsf{fut}(\varphi)}{2} \rceil + 1$ clocks and $((2\lceil \frac{k \cdot \mathsf{fut}(\varphi)}{2} \rceil + 1)^m + 1)(2 \cdot 4^n + 1)$ states that accepts the language of φ.*

5 Deterministic Timed Automata

We show that the automata constructed in Section 4 can be determinized. The separation of the construction to deterministic proposition monitors and nondeterministic DTA makes this part possible. We use a variant of Piterman's version of Safra's determinization [23] to determinize DTA. The differences are mostly syntactic, taking into account the 'asynchronicity' of transitions from a set of states. We assume that every transition (q, g, o, q') of a DTA 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 $(t, t') \subseteq I(q')$. The DTA from Section 4 satisfy this condition. We also assume that the DTA takes infinitely many steps and that its acceptance set $\mathcal{F}$ contains exactly one set. It is simple to modify the automaton to an automaton that takes infinitely many steps by adding (or reusing) a clock that keeps resetting itself and taking

[5] Due to the lack of space, we restrict attention to intervals of the form $[b, b]$, (a, b) and (a, ∞) and include a full treatment of all other types of intervals in the full version of the paper.

a transition whenever this clock reaches some value. Converting an automaton where $|\mathcal{F}| > 1$ to an automaton where $|\mathcal{F}| = 1$ is standard. We make this assumption solely to simplify presentation.

Let $B = \langle \Sigma, \Gamma, Q, \mathcal{C}, \gamma, I, \Delta, q_0, \mathcal{F} \rangle$ be a DTA satisfying these conditions. We construct an 'equivalent' DTA D. The construction is based on the subset construction [25]. Thus, every state of D is associated with a set of states of B. A state of D associated with $Q' \subseteq Q$ follows 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 $\bigwedge_{(q,g,o,q') \in \Delta'} \neg g$. A set of runs of B that end in states in Q' can be extended in three different ways: 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 $succ(Q'', T)$ be $stay(T) \cap Q'' \cup \{q \mid \exists (q', g, o, q) \in T \cap \Delta(Q'')\}$. For every T as above, a state of D associated with Q' can take a T-transition with guard $g(T)$:

$$I(stay(T)) \wedge \overline{g}(\Delta(stay(T))) \wedge g(T \cap \Delta) \wedge \overline{g}(\Delta(move(T)) \setminus T) \wedge \\ \overline{I}(deadend(Q', T)) \wedge \overline{g}(\Delta(deadend(Q', T)))$$

That is, states whose run is extended by staying in the same state contribute their invariants and the negation of transitions exiting them; transitions that are crossed contribute their guards and the negation of transitions that are not crossed; and states whose run is going to end contribute the negation of their invariants and the guards of transitions exiting them. The case when both $deadend(T)$ and $T \cap \Delta$ are empty is not interesting as all runs are extended.

Definition 4. [23] *A compact Safra tree t over Q is $\langle N, M, 1, p, l, e, f \rangle$, where the components of t are as follows. Let $|Q| = n$. (a) $N \subseteq [n]$ is a set of nodes. (b) $1 \in N$ is the root node. (c) $p : N \to N$ is the parent function. (d) $l : N \to 2^Q$ is a labeling of the nodes with subsets of Q. The label of every node is a proper superset of the union of the labels of its sons. The labels of two siblings are disjoint. (e) $e, f \in [n+1]$ are used to define the parity acceptance conditions. For a tree t, let $set(t) = \bigcup_{i \in [n]} l(i)$ be the set of states that label at least one node in t.*

The deterministic DTA is $D = \langle \Sigma, \{o\}, S, \mathcal{C}, \gamma', I', \Delta', s_0, \alpha \rangle$, where components are defined as follows. Differences from the construction in [23] are in bold.

- S is the set of compact Safra trees over Q.
- s_0 is the tree with a single node 1 labeled $\{q_0\}$, where $e = 2$ and $f = 1$.
- The parity acceptance condition $\alpha = \langle F_0, \ldots, F_{2n-1} \rangle$, where $F_{2i} = \{s \in S \mid f = i + 1 \text{ and } e > f\}$ and $F_{2i+1} = \{s \in S \mid e = i + 2 \text{ and } f \geq e\}$.
- **For every state $s \in S$ we have $\gamma'(s) = o$ and $I'(s) = I(set(s))$.**

– For every tree $s \in S$ and **every set $\boldsymbol{T \subseteq set(s) \cup \Delta(set(s))}$ we add to $\boldsymbol{\Delta'}$ the transition $\boldsymbol{(s, g(T), o, s')}$** where s' is obtained from s using the following transformations.
  1. For every node v with label Q' replace Q' by $\boldsymbol{succ(Q', T)}$.
  2. For every node v with label Q' such that $Q' \cap \alpha \neq \emptyset$, create a new son $v' \notin N$ of v. Set its label to $Q' \cap \alpha$. Set its name to the minimal value greater than all used names. We may have to use temporarily names in the range $[(n+1)..(2n)]$.
  3. For every node v with label Q' and state $q \in Q'$ such that q belongs also to some sibling v' of v such that $M(v') < M(v)$, remove q from the label of v and all its descendants.
  4. For every node v whose label is equal to the union of the labels of its children, remove all descendants of v. Call such nodes *green*. Set f to the minimum of $n+1$ and all green nodes. Notice that no node in $[(n+1)..(2n)]$ can be green.
  5. Remove all nodes with empty labels. Set e to the minimum of $n+1$ and all nodes removed during all stages of the transformation.
  6. Let Z denote the set of nodes removed during all previous stages of the transformation. For every node v let $rem(v)$ be $|\{v' \in Z \mid M(v') < M(v)\}|$. That is, we count how many nodes that are smaller than v are removed during the transformation. For every node v such that $l(v) \neq \emptyset$ we change v to $M(v) - rem(v)$.

Theorem 1. *For every deterministic timed automaton A, we have $A \otimes D$ is deterministic and $L(A \otimes D) = L(A \otimes B)$.*

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

We note that the double exponent in the number of unbounded temporal operators is unavoidable, as follows from the same for LTL.

6 Conclusions

We developed a novel construction for translating full MTL to timed automata, under bounded variability assumptions. Our construction provides a unified framework for model checking, runtime monitoring, and controller synthesis, and offers an alternative translation that improves exponentially on the complexity of securing a deterministic timed automaton, avoiding a doubly exponential number of clocks.

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

– 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 monitoring 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 and evaluate it in the context of model checking, runtime monitoring, and controller synthesis.

Acknowledgements We would like to thank Dana Fisman and Oded Maler for their insightful suggestions that helped us improve the clarity of the manuscript.

References

1. R. Alur. Timed automata. In *CAV*, LNCS 1633, pp. 8–22. Springer, 1999.
2. E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *JACM*, 49(2):172–206, 2002.
3. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126(2):183–236, 1994.
4. R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *JACM*, 43(1):116–146, 1996.
5. R. Alur and T.A. Henzinger. Logics and models of real time: a survey. In *Real Time: Theory in Practice*, LNCS 600, pp. 74–106. Springer, 1992.
6. E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *IFAC SSSC*, pp. 469–474. Elsevier, 1998.
7. C. Baier, N. Bertrand, P. Bouyer and T. Brihaye. When are Timed Automata Determinizable? In *ICALP*, LNCS 5556, pp. 43–54. Springer, 2009.
8. P. Bouyer, L. Bozzelli and F. Chevalier. Controller Synthesis for MTL Specifications. In *CONCUR*, LNCS 4137, 450–464. Springer, 2006.
9. 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.
10. L. Doyen, G. Geeraerts, J.-F. Raskin, and J. Reichert. Realizability of real-time logics. In *FORMATS*, LNCS 5813, Springer, 133–148, 2009.
11. K. Havelund and G. Rosu. Efficient monitoring of safety properties. *STTT*, 2004.
12. T.A. Henzinger. It's about time. In *Concur*, LNCS 1466, pp. 439–454. Springer, 1998.
13. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *I&C*, 111:193–244, 1994.
14. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.
15. K. G. Larsen, P. Petterson, and W. Yi. UPPAAL: Status & developments. In *CAV*, LNCS 1254, pp. 456–459. Springer, 1997.
16. O. Maler, D. Nickovic, and A. Pnueli. Real time temporal logic: Past, present, future. In *FORMATS*, LNCS 3829, pp. 2–16. springer, 2005.
17. O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *FORMATS*, LNCS 4202, pp. 274–289. Springer, 2006.
18. O. Maler, D. Nickovic, and A. Pnueli. On synthesizing controllers from bounded-response properties. In *CAV*, LNCS 4590, pp. 95–107. Springer, 2007.
19. D. Monniaux. A quantifier elimination algorithm for linear real arithmetic. In *LPAR*, LNCS 5330, pp. 243–257. Springer, 2008.
20. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Specification*. Springer, 1991.
21. O.Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS*, LNCS 900. Springer, 1995.
22. J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS*, pp. 188–197, 2005.
23. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *LMCS*, 3(3):5, 2007.
24. A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module. In *POPL*, 179–190,1989.
25. M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of R&D*, 3:115–125, 1959.
26. S. Safra. On the complexity of ω-automata. In *FOCS*, pp. 319–327, 1988.
27. M.Y. Vardi and P. Wolper. An Automata-theoretic Approach to Automatic Program Verification, In *LICS*, pp. 322–331, 1986.
28. T. Wilke. Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata, In *FTRTFT*, LNCS 863, pp. 694–715. Springer, 1994.
29. S. Yovine. Kronos: A verification tool for real-time systems. *STTT*, 1(1–2):123–133, 1997.