Acta Informatica (2014) 51:165–192
DOI 10.1007/s00236-013-0189-z

ORIGINAL ARTICLE

Efficient controller synthesis for a fragment of $\mathbf{MTL}_{0,\infty}$

Peter Bulychev · Alexandre David · Kim G. Larsen · Guangyuan Li

Received: 2 February 2013 / Accepted: 8 November 2013 / Published online: 17 December 2013

Abstract In this paper we offer an efficient controller synthesis algorithm for assume-guarantee specifications of the form $\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \rightarrow \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_m$. Here, $\{\varphi_i, \psi_j\}$ are all safety-$\mathrm{MTL}_{0,\infty}$ properties, where the sub-formulas $\{\varphi_i\}$ are supposed to specify assumptions of the environment and the sub-formulas $\{\psi_j\}$ are specifying requirements to be guaranteed by the controller. Our synthesis method exploits the engine of UPPAAL-TIGA and the novel translation of safety- and co-safety-$\mathrm{MTL}_{0,\infty}$ properties into under-approximating, deterministic timed automata. Our approach avoids determinization of Büchi automata, which is the main obstacle for the practical applicability of controller synthesis for linear-time specifications. The experiments demonstrate that the chosen specification formalism is expressive enough to specify complex behaviors. The proposed approach

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n 601148 (Cassting). This publication reflects only the authors views and the Union is not liable for any use that may be made of the information contained herein. Also, the paper is supported by the Danish National Research Foundation, the National Natural Science Foundation of China (Grant No. 61061130541) for the Danish-Chinese Center for Cyber Physical Systems and VKR Center of Excellence MT-LAB.

P. Bulychev (✉) · A. David · K. G. Larsen
CISS, CS, Aalborg University, Ålborg, Denmark
e-mail: peter.bulychev@gmail.com

A. David
e-mail: adavid@cs.aau.dk

K. G. Larsen
e-mail: kgl@cs.aau.dk

G. Li
State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing, China
e-mail: ligy@ios.ac.cn

Present address:
P. Bulychev
Google Inc., Zurich, Switzerland, CA, USA

is sound but not complete. However, it successfully produced solutions for all the experiments. Additionally we compared our tool with Acacia+ and Unbeast, state-of-the-art LTL synthesis tools; and our tool demonstrated better timing results, when we applied both tools to the analogous specifications.

1 Introduction

Automatic controller synthesis is concerned with the algorithmic construction of a control strategy, that will ensure a given behavioural specification to be satisfied regardless of the input provided by an environment. This problem was first stated in a discrete time setting by Church in 1962 in [14] and then theoretically solved for various specification formalisms in [11] and later works [10,19,24,28,31,32].

The synthesis problem is computationally harder for linear time logics than the satisfiability and model-checking problems, and has for this reason been considered to be intractable for a long time. The main problem was that the first synthesis approaches involve the determinization of Büchi automata, which is a computationally hard problem. However, the synthesis problem has recently gained in practical performance with the development of the so-called Safraless synthesis algorithms [23] that avoid the Büchi determinization phase. This approach has been strengthened by using the bounded synthesis [33] and the antichain-based algorithm in [20] that have resulted in the tools ACACIA and ACACIA+ that demonstrate good performance for the reasonable sized formulas arising in practice [21]. Another direction of research tries to restrict the specification formalisms so that the synthesis problem will be easier to solve [7,30]. We combine both directions, i.e. restrict ourselves to a formalism that does not require Safra's procedure.

In the current paper we study the synthesis problem in a real-time setting. We focus our attention to the Metric Interval Temporal Logic (MITL) [3], a formalism which has proved to be useful for specifying real-time systems [1]. Unfortunately, the MITL synthesis problem is undecidable in general [17], but there are sub-classes that render the synthesis problem to be decidable [10,26]. The main challenge for the real-time controller synthesis is that the Safraless approach is not always applicable for the timed case since determinization is not possible in general. Thus it makes sense to completely avoid the automata determinization phase.

In the current paper we push the boundaries of MITL synthesis forward to an important class of assume-guarantee properties, i.e. properties of the form $\Psi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \rightarrow \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_m$. Here the implicants $\{\varphi_i\}$ are typically used to specify assumptions of the behaviour of an environment, and the implied part $\{\psi_j\}$ is used to specify the requirements to the controller. We assume that every φ_i and ψ_j belong to safety-MTL$_{0,\infty}$, being the positive fragment of MITL where all until operators should have only upper bounds, and all release operators can have either upper or lower bound. Still, the safety-MTL$_{0,\infty}$ fragment allows us to express important bounded-response properties, e.g. $\Box(a \rightarrow \Diamond_{\leq 1} b)$ that tells that every a should be followed by b within 1 time unit.

As we show in a number of case-studies such assume-guarantee properties can be used to express many useful real-time specifications; and our implementation can handle these specifications. At the same time this formalism lives close to the edge of undecidability, as e.g. allowing the lower bounds for the until operator makes the synthesis problem undecidable [10]. However, we believe that our approach can be extended to the specifications being arbitrary Boolean combinations of safety-MTL$_{0,\infty}$ formulas (not only implications).

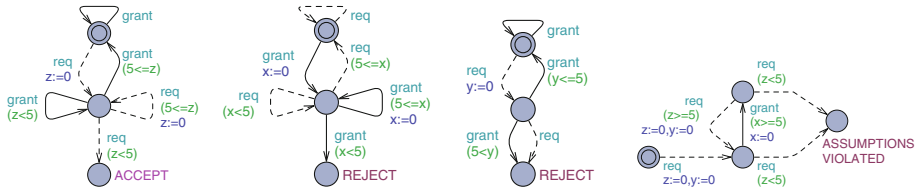

Fig. 1 Deterministic monitors for $\neg\Box(req \rightarrow \widehat{\Box}_{<5}\neg req)$, $\Box(grant \rightarrow \widehat{\Box}_{<5}\neg grant)$ and $\Box(req \rightarrow \neg req\,\widehat{\mathrm{U}}_{\leq 5} grant)$, and the winning strategy (from left to right)

Our approach translates the overall specification Ψ to a network of timed game automata—one automaton for every requirement ψ_j and one automaton for the negation of every assumption φ_i. A central contribution is the extension of the translation presented in [12], with the advantage that the resulting automata contain no Büchi acceptance conditions and are *deterministic by construction*. Hence, there is no need for a determinization phase. However, it is not always possible to translate $\mathrm{MTL}_{0,\infty}$ properties into language-equivalent deterministic automata,[1] and in this case we produce a deterministic under-approximation. Thus our approach is sound but not complete, i.e. it may fail to generate a strategy even if the specification is realizable. As we shall see, this theoretical incompleteness does not show up in our case studies.

It now remains to use UPPAAL- TIGA [5] to synthesize a control strategy for the resulting timed game fulfilling all the requirements $\{\psi_j\}$ or violating at least one of the assumptions $\{\varphi_i\}$. Additionally, the synthesized strategy should avoid generating Zeno behaviour (playing infinite number of actions in a finite amount of time). We do this by either forcing UPPAAL-TIGAto generate non-Zeno strategy using Büchi winning condition; or by proving that no Zeno strategy will be winning for the controller. Finally, having synthesized a winning strategy, it may be transformed into executable code.

As an illustration, consider the specification:[2]

$$\Box(req \rightarrow \widehat{\Box}_{<5}\neg req) \rightarrow \Box(grant \rightarrow \widehat{\Box}_{<5}\neg grant) \wedge \Box(req \rightarrow (\neg req)\widehat{\mathrm{U}}_{\leq 5}(grant))$$

This specification says that every request (req) should be served ($grant$) within 5 time-units under the assumption that the environment does not make repeated requests with less than a separation of 5 time-units. Also, the specification forbids grants to be issued too frequently—again a minimum time-separation of 5 time-units is required. Figure 1 contains the network of deterministic timed game automata generated by our approach for this specification, together with a winning strategy for this network synthesized by UPPAAL- TIGA. Here, the dashed transitions are uncontrollable, and in the strategy the controller always proposes to take the earliest enabled controllable transition. Although some formulas are defined over single actions, the corresponding automata use both actions since $\neg req$ renders to $grant$ (and the other way round) under the assumption that only these two actions are available.

1.1 Related work

The paper [27] is the first work that proposed the algorithms for the controller synthesis based on timed games. In this work the game rules (i.e. what controller and environment can

[1] For instance, there is no equivalent deterministic Timed Automata for $\mathrm{MTL}_{0,\infty}$ property $\Diamond_{\leq 1}(p \wedge \Box_{\leq 1}(\neg r) \wedge \Diamond_{\leq 1}(q))$. This can be proved by adapting the proof of [25] that $\mathrm{MITL}_{[a,b]}$ formula $\Box_{\leq 1}(p \rightarrow \Diamond_{[1,2]}(\neg q))$ is not determinizable.

[2] As we will define in Sect. 2, $\widehat{\Box}$ and $\widehat{\mathrm{U}}$ are modifications of the "classical" globally and until operators that do not take into account the first observation of a trace.

potentially do) were defined as timed game automata. The winning goals (i.e. what controller should achieve) were expressed as Rabin conditions. This differs from our setup since we use linear-time specifications to define both parts. Later the authors of [13] restricted their attention to timed games with reachability and safety properties. They proposed an efficient symbolic on-the-fly algorithm that has been implemented in the UPPAAL- TIGA tool [5].

The paper [26] was the first work that studied the controller synthesis problem for the case when the specification itself is given as a linear-time temporal formula and a system works in real time. This work proposes an algorithm for the synthesis of the properties of the form $\Box\varphi$, where φ is a safety-MITL$_{\leq}$ property that can also contain past formulas. Our approach works well for assume-guarantee properties and it might be hard to fit such properties into the logic of [26]. Additionally, [26] makes the bounded variability assumption, i.e. it assumes that the number of times a signal changes per one time unit is bounded by some known constant k. This might be too strict in the dense time setting and we do not make this assumption.

The paper [10] and [17] present a series of decidability and undecidability results for the synthesis and realizability for different fragments of real-time logics MTL, ECL and LTL$_{\lhd}$. However it is an open question whether the synthesis problem for our fragment of MTL is decidable or not. If we limit the resources of the controller (i.e. the number of clocks it can have and the maximum constants), then the problem is decidable according to [10].

The paper [16] proposes the Safraless procedure for the synthesis of LTL$_{\lhd}$ properties.

The translation procedure of [16] involves the solution of a timed game with $k^{2^{|\varphi|}}$ locations, where k depends on the specification itself and can be relatively large.

Outline. The rest of the paper is organized as following. Section 2 provides all the necessary definitions. In Sect. 3 we formally state the problem being solved. Section 4 describes the novel translation of safety- and co-safety-MTL$_{0,\infty}$ properties into exact non-deterministic monitoring Timed Automata and under-approximation deterministic Timed Automata. Section 5 describes how we use UPPAAL- TIGA to solve the synthesis games. Section 6 contains the case studies.

2 Definitions and problem statement

2.1 Timed words and MTL$_{0,\infty}$ logic

A timed word over a finite set of actions Σ is a finite or infinite sequence of time points and actions $(t_1, a_1)(t_2, a_2)(t_3, a_3)\ldots$, where for every i we have $a_i \in \Sigma$, $t_i \in \mathbb{R}_{\geq 0}$ and for any non-last i we have $t_{i+1} \geq t_i$. Let $Runs(\Sigma)$ denote the set of all *finite* timed words over Σ and $InfRuns(\Sigma)$ the set of all *infinite* timed words over Σ.

A timed word is called Zeno iff it contains infinitely many discrete actions during a finite amount of time. Certainly, the controllers that produce Zeno timed words are not implementable in real life. In our modeling formalism Zeno behavior can also be forced by the environment (if it emits uncontrollable actions infinitely often). In order to distinguish these two situations we define Zeno behavior over sets of actions. A timed word $(t_1, a_1)(t_2, a_2)(t_3, a_3)\ldots$ is called Zeno for $\Sigma' \subseteq \Sigma$, if it is time-bounded (i.e. $t_i < t$ for some t and for all i) and there are infinitely many actions from Σ' in it (i.e. for any j there exists $i > j$ such that $a_i \in \Sigma'$). Let $Zeno(\Sigma')$ denote the set of all infinite timed words that are Zeno for Σ'.

Definition 1 An MTL$_{0,\infty}$ formula φ over actions Σ is defined by the grammar

$$\varphi ::= true \mid a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \mathrm{U}_{\sim d}\, \varphi_2 \mid \varphi_1 \widehat{\mathrm{U}}_{\sim d}\, \varphi_2$$

where $a \in \Sigma$, $\sim \in \{<, \leq, \geq, >\}$ and $d \in \mathbb{N}$.

The common abbreviations are: $false = \neg true$, $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 = (\neg\varphi_1) \vee \varphi_2$, $\varphi_1 \mathrm{R}_{\sim d}\, \varphi_2 = \neg(\neg\varphi_1 \mathrm{U}_{\sim d}\, \neg\varphi_2)$, $\varphi_1 \widehat{\mathrm{R}}_{\sim d}\, \varphi_2 = \neg(\neg\varphi_1 \widehat{\mathrm{U}}_{\sim d}\, \neg\varphi_2)$, $\Diamond_{\sim d}\, \varphi = true\, \mathrm{U}_{\sim d}\, \varphi$, $\widehat{\Diamond}_{\sim d}\, \varphi = true\, \widehat{\mathrm{U}}_{\sim d}\, \varphi$, $\Box_{\sim d}\, \varphi = false\, \mathrm{R}_{\sim d}\, \varphi$, and $\widehat{\Box}_{\sim d}\, \varphi = false\, \widehat{\mathrm{R}}_{\sim d}\, \varphi$.

The temporal operators R and $\widehat{\mathrm{R}}$ are called *release* operators, and U and $\widehat{\mathrm{U}}$ are called *until* operators. The release and until operators are called bounded if they have a bound "$< d$" or "$\leq d$", otherwise unbounded. We will omit writing a bound when the bound is "≥ 0" (e.g. write $\varphi_1 \mathrm{U} \varphi_2$ instead of $\varphi_1 \mathrm{U}_{\geq 0}\, \varphi_2$).

Safety-MTL$_{0,\infty}$ (co-safety-MTL$_{0,\infty}$) is the fragment of MTL$_{0,\infty}$ that consists of all the formulas in the positive normal form that do not use unbounded until (release, correspondingly) operator [29]. For instance, $\Box \Diamond_{\leq 1} a$ is in safety-MTL$_{0,\infty}$, $\Diamond \Box_{\leq 1} a$ is in co-safety-MTL$_{0,\infty}$, $\Box_{\leq 1} \Diamond_{\leq 1} a$ is both in safety- and co-safety-MTL$_{0,\infty}$, $\Box\Diamond a$ is in none of them. It is easy to see that an MTL$_{0,\infty}$-formula φ is in safety-MTL$_{0,\infty}$ iff its negation is in co-safety-MTL$_{0,\infty}$.

The semantics of MTL$_{0,\infty}$ is defined over *infinite* timed words. Let w^i be the i-th suffix of the timed word w. For a given infinite timed word $w = (t_1, a_1)(t_2, a_2)(t_3, a_3) \ldots$ and an MTL$_{0,\infty}$ formula φ, the satisfaction relation $w^i \models \varphi$ is defined inductively:

1. $w^i \models true$
2. $w^i \models a$ iff $a_i = a$
3. $w^i \models \neg\varphi$ iff $w^i \not\models \varphi$
4. $w^i \models \bigcirc\varphi$ iff $w^{i+1} \models \varphi$
5. $w^i \models \varphi_1 \wedge \varphi_2$ iff $w^i \models \varphi_1$ and $w^i \models \varphi_2$
6. $w^i \models \varphi_1 \mathrm{U}_{\sim d}\, \varphi_2$ where $\sim \in \{<, \leq, \geq, >\}$ iff there exists j such that $j \geq i$, $w^j \models \varphi_2$, $t_j - t_i \sim d$, and $w^k \models \varphi_1$ for all k with $i \leq k < j$
7. $w^i \models \varphi_1 \widehat{\mathrm{U}}_{\sim d}\, \varphi_2$ where $\sim \in \{<, \leq, \geq, >\}$ iff there exists j such that $j > i$, $w^j \models \varphi_2$, $t_j - t_i \sim d$, and $w^k \models \varphi_1$ for all k with $i < k < j$

An infinite timed word w satisfies an MTL$_{0,\infty}$-formula φ iff $w^1 \models \varphi$. The language $\mathcal{L}(\varphi)$ of φ is the set of all infinite timed words that satisfy φ.

2.2 MTL$_{0,\infty}$ synthesis problem

A synthesis problem is defined by a triple $(\varphi, \Sigma_c, \Sigma_u)$, where Σ_c and Σ_u are disjoint sets of elements (that are called controllable and uncontrollable actions), and φ is an MTL$_{0,\infty}$ formula over $\Sigma_c \cup \Sigma_u$. This triple defines the rules of the game between the controller and the environment. The game is played in rounds, and at each round the controller proposes to play some controllable action after some delay, and then the environment can let the controller do this, or it can overtake the controller's choice with a smaller or equal delay and some uncontrollable action. The behavior of the controller is determined by a strategy, a function that during the course of the game constantly gives information as to what the controller should do in order to satisfy the specification.

More formally, a strategy of the controller in $(\varphi, \Sigma_c, \Sigma_u)$ is a function f that maps the finite timed words from $Runs(\Sigma_c \cup \Sigma_u)$ to the pairs (τ, a), where $\tau \in \mathbb{R}_{\geq 0}$ and $a \in \Sigma_c$. An *infinite* timed word $(t_1, a_1)(t_2, a_2)(t_3, a_3) \ldots$ is consistent with the strategy f, if for any i, if $f((t_1, a_1)(t_2, a_2) \ldots (t_i, a_i)) = (\tau, a)$, then either $(\tau, a) = (t_{i+1} - t_i, a_{i+1})$, or $a_{i+1} \in \Sigma_u$ and $t_{i+1} - t_i \leq \tau$. Let $Outcomes(f, \Sigma_c, \Sigma_u) \subseteq InfRuns(\Sigma_c \cup \Sigma_u)$ be the set of all infinite timed words over $\Sigma_c \cup \Sigma_u$ that are consistent with the strategy f.

Since the synthesis games are played in dense real time, we should prevent the controller from blocking time by using the Zeno strategies. We assume that a Zeno run is losing for the controller iff there are infinitely many uncontrollable actions among it (i.e. the controller does

not block time by itself). This means that if the environment emits infinitely many actions among a finite amount of time, then it loses even if the controller also emits infinitely many actions.

We come to the following definition:

Definition 2 A controller's strategy f is called *a solution* for the synthesis problem $(\varphi, \Sigma_c, \Sigma_u)$, iff for any run $\omega \in Outcomes(f, \Sigma_c, \Sigma_u) \backslash Zeno(\Sigma_u)$ we have $\omega \in (\mathcal{L}(\varphi) \backslash Zeno(\Sigma_c))$

Obviously, for the practical application of the automatic synthesis method, a controller's strategy should be implementable and thus it should have a finite representation. In the current paper we assume that a strategy in the synthesis problem is defined by a deterministic Timed Automata, i.e. a controller has finitely many clocks that can record time passed since some events, and finitely many control states that define the reaction to the input events.

2.3 Timed automata

Let X be a set of real-valued variables called clocks. A clock bound over X has the form $x \sim n$ where $x \in X$, $\sim \in \{<, \leq, \geq, >\}$ and $n \in \mathbb{Z}_{\geq 0}$. We denote the set of all possible clock bounds over X by $\mathcal{B}(X)$, and $\mathcal{F}(X)$ is the set of all Boolean formulas over $\mathcal{B}(X)$ (including conjunctions and disjunctions). A valuation over X is a an element of $\mathbb{R}^X_{\geq 0}$, i.e. it is a function $v : X \to \mathbb{R}_{\geq 0}$. We let $\overline{0}$ be the valuation that assigns 0 to any clock from X. For a given valuation v, clock set $Y \subseteq X$ and real number $\tau \in \mathbb{R}_{\geq 0}$ we let $v + \tau$ to be the valuation such that $(v + \tau)(x) = v(x) + \tau$ for every clock $x \in X$; and $v[Y]$ is equal to the valuation such that $v[Y](x) = 0$ if $x \in Y$ and $v[Y](x) = v(x)$ otherwise.

Definition 3 [2] A *Timed Automaton* (TA) over actions Σ is a tuple (L, l_0, X, E) where:

- L is a finite set of locations,
- l_0 is the initial location,
- X is a finite set of clocks,
- $E \subseteq L \times \Sigma \times \mathcal{F}(X) \times 2^{|X|} \times L$ is the finite set of edges.

The semantics of TA is defined by Labeled Transition System (LTS) $(S, s_0, \to)$. A set of states $L \times \mathbb{R}^X_{\geq 0}$ of a TA consists of pairs of locations and valuations over X. The initial state is $(l_0, \overline{0})$. There exists a *delay* transition $(l, v_1) \xrightarrow{\tau} (l, v_2)$, iff $\tau \in \mathbb{R}_{\geq 0}$ and $v_2 = v_1 + \tau$. There exists a *discrete* transition $(l_1, v_1) \xrightarrow{a} (l_2, v_2)$ if there exists a transition (l_1, a, g, Y, l_2) such that $v_1 \models g$ and $v_2 = v_1[Y]$. In the latter case we say that a transition e is *enabled* in the state (l_1, v_1).

A TA is called *total* for a set of actions $\Sigma' \subseteq \Sigma$ if for any action $a \in \Sigma'$ and for any state $s \in S$ there exists at least one state s' such that $s \xrightarrow{a} s'$. A TA is called *deterministic* iff not more than one such state s' exists for any pair of s and a.

A run of a TA is an infinite sequence of alternating delay and discrete transitions $s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{a_1} s_2 \xrightarrow{\tau_2} \cdots$.

A timed word $w = (t_1, a_1)(t_2, a_2)(t_3, a_3) \ldots$ over Σ is accepted by timed automaton A iff there is an infinite sequence of states $s_0, s_1, s_2, \ldots$ such that s_0 is the initial state of A and $s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{a_1} s_2 \xrightarrow{\tau_2} \ldots$ is a run of A, where $\tau_i = t_i - t_{i-1}$ for every $i > 1$ and $t_1 = \tau_1$. We use $\mathcal{L}(A)$ to denote the set of all timed words that are accepted by A.

We use TA to define synthesis game arenas (where both players have freedom to choose time delays and actions); and to define the controller's winning strategies in which the controller always takes the earliest enabled controllable transition. The latter TA define winning strategies in the synthesis problems as the following:

Definition 4 Let $(\varphi, \Sigma_c, \Sigma_u)$ be a synthesis problem and let A be a total for Σ_u and deterministic TA. We say that A implements the strategy f_A in $(\varphi, \Sigma_c, \Sigma_u)$ defined in the following way. Let $\omega = (t_1, a_1)(t_2, a_2) \dots (t_n, a_n)$ be an arbitrary finite timed word over $\Sigma_c \cup \Sigma_u$. Let $s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{a_1} s_2 \dots s_{2n}$ be a unique run of A on ω. Let τ_{n+1} be the minimal delay such that there exist transitions $s_{2n} \xrightarrow{\tau_{n+1}} s_{2n+1}$ and $s_{2n+1} \xrightarrow{a_{n+1}} s_{2n+2}$ for some *controllable* action $a_{n+1} \in \Sigma_c$. Then we define the value of the strategy f_A on ω as $f_A(\omega) = (\tau_{n+1}, a_{n+1})$.

3 Problem statement

Consider the sets Σ_c and Σ_u of controllable and uncontrollable actions. Consider an $\mathrm{MTL}_{0,\infty}$ property φ over $\Sigma_c \cup \Sigma_u$ such that

$$\Psi \equiv \varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n \rightarrow \psi_1 \wedge \psi_2 \wedge \dots \wedge \psi_m$$

where all φ_i and ψ_j are safety-$\mathrm{MTL}_{0,\infty}$ properties. Our goal is to construct a TA A such that A implements a solution f_A for the synthesis problem $(\Psi, \Sigma_c, \Sigma_u)$.

4 From safety- and co-safety-$\mathrm{MTL}_{0,\infty}$ to Timed Automata

In our paper [12] we presented a procedure to translate $\mathrm{MITL}_{\leq}$ formulas, the bounded fragment of $\mathrm{MTL}_{0,\infty}$, into Timed Automata. For a given formula, the procedure can build language-equivalent nondeterministic timed automata, as well as deterministic under- and over-approximations.

In the current work we will extend the procedure of [12] to the case of safety and co-safety $\mathrm{MTL}_{0,\infty}$. Basically, we use the classical tableau-based translation from linear-time temporal formulas into Büchi automata, with two important modifications. First, we avoid using Büchi acceptance condition because we consider safety and co-safety fragments of $\mathrm{MTL}_{0,\infty}$ on non-Zeno runs only (e.g. we cannot express $\Box\Diamond a$ that certainly needs Büchi acceptance condition). Second, in order to handle timed properties, we introduce an auxiliary clock for each subformula of the form $\varphi_1 \mathrm{U}_{\sim d} \varphi_2$, $\varphi_1 \mathrm{R}_{\sim d} \varphi_2$, $\varphi_1 \widehat{\mathrm{U}}_{\sim d} \varphi_2$ or $\varphi_1 \widehat{\mathrm{R}}_{\sim d} \varphi_2$(where $\sim \in \{<, \leq, \geq, >\}$ and $\sim d$ is not ≥ 0). When a timed formula, for example, $\varphi_1 \mathrm{U}_{\leq d} \varphi_2$ is expected to be satisfied, the automaton will reset the corresponding clock x, rewrite this formula into $\varphi_1 \mathrm{U}_{\leq d-x} \varphi_2$ and in future observations compare the value of x with d, and check if the promise $\varphi_1 \mathrm{U}_{\leq d-x} \varphi_2$ has been fulfilled within the time bound.

In the rest of this section, we assume that φ is an $\mathrm{MTL}_{0,\infty}$ formula over Σ and has been transformed into positive normal form, where the negation operator $(\neg)$ is not allowed ($\neg true$ is replaced by $false$ and $\neg a$ is replaced by $\bigvee_{b \in \Sigma \setminus \{a\}} b$ when a is an action in Σ). For the sake of simplicity, we also make an assumption that all temporal operators occurring in φ are included in $\{\mathrm{U}_{\leq d}, \mathrm{R}_{\leq d}, \mathrm{U}_{\geq d}, \mathrm{R}_{\geq d}\}$.

4.1 Closures and extended formulas

We use $\mathrm{Sub}(\varphi)$ to denote all the sub-formulas of φ. For each $\varphi_1 \mathrm{U}_{\leq d} \varphi_2 \in \mathrm{Sub}(\varphi)$, we assign a clock $x_{(\varphi_1 \mathrm{U}_{\leq d} \varphi_2)}$ to it. Let $X_{U\leq}$ be the set of all $\mathrm{U}_{\leq d}$-clocks, i.e. $\{x_{(\varphi_1 \mathrm{U}_{\leq d} \varphi_2)} \mid \varphi_1 \mathrm{U}_{\leq d} \varphi_2 \in$

Sub(φ)}. Similarly we assign the clocks to the other until and release operators, and define $X_{U\geq}$, $X_{R\leq}$ and $X_{R\geq}$. We do not assign clocks to formulas $\varphi_1 \mathrm{U} \varphi_2$ and $\varphi_1 \mathrm{R} \varphi_2$, therefore we always assume that $d > 0$ when we write $\mathrm{U}_{\geq d}$ or $\mathrm{R}_{\geq d}$ in this section.

The set of *basic formulas* for φ, written as BF(φ), is a finite set defined by the following rules:

1. If $\bigcirc\varphi_1 \in \mathrm{Sub}(\varphi)$, then $\varphi_1 \in \mathrm{BF}(\varphi)$
2. If $\varphi_1 \mathrm{U} \varphi_2 \in \mathrm{Sub}(\varphi)$ or $\varphi_1 \mathrm{U}_{\geq d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$, then $\varphi_1 \mathrm{U} \varphi_2 \in \mathrm{BF}(\varphi)$
3. If $\varphi_1 \mathrm{R} \varphi_2 \in \mathrm{Sub}(\varphi)$ or $\varphi_1 \mathrm{R}_{\geq d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$, then $\varphi_1 \mathrm{R} \varphi_2 \in \mathrm{BF}(\varphi)$
4. If $\varphi_1 \mathrm{U}_{\sim d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$ and x is the clock assigned to $\varphi_1 \mathrm{U}_{\sim d}\, \varphi_2$, then $\varphi_1 \mathrm{U}_{\sim d - x}\, \varphi_2, x \sim d \in \mathrm{BF}(\varphi)$, where $\sim \in \{\leq, \geq\}$.
5. If $\varphi_1 \mathrm{R}_{\sim d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$ and x is the clock assigned to $\varphi_1 \mathrm{R}_{\sim d}\, \varphi_2$, then $\varphi_1 \mathrm{R}_{\sim d - x}\, \varphi_2, \overline{x \sim d} \in \mathrm{BF}(\varphi)$, where $\sim \in \{\leq, \geq\}$ and $\overline{x \sim d}$ is the negation of $x \sim d$ (for example, $\overline{x \leq d} = x > d$ and $\overline{x \geq d} = x < d$)

The meaning of $\mathrm{U}_{\leq d-x}$, $\mathrm{U}_{\leq d-x}$, $\mathrm{R}_{\leq d-x}$ and $\mathrm{R}_{\geq d-x}$ will be given in Definition 5.

Since a conjunction of basic formulas can be regarded as a subset of BF(φ), for simplicity, we use $2^{\mathrm{BF}(\varphi)}$ for the set of all subsets of BF(φ) as well as the set of all conjunctive formulas over BF(φ). Because a conjunction with zero conjuncts is true, so $true \in 2^{\mathrm{BF}(\varphi)}$.

We define CL(φ), the closure of φ, to be the set of all positive Boolean combinations (i.e., without negation) over BF(φ). Obviously, CL(φ) has only finitely many different non-equivalent formulas.

For a clock $x \in X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq}$, we use $rst(x)$ to represent that x will be reset at the current step. Similarly, for $x \in X_{\mathrm{U}\leq} \cup X_{\mathrm{R}\geq}$, we use $unch(x)$ to represent that x will not be reset at the current step. Now we defined Ext(φ), the set of extended formulas for φ, with the following rules:

1. $\mathrm{Sub}(\varphi) \subseteq \mathrm{Ext}(\varphi)$
2. If $\phi \in \mathrm{CL}(\varphi)$, then $\phi, \bigcirc\phi \in \mathrm{Ext}(\varphi)$
3. If $x \in X_{\mathrm{U}\leq}$ or $x \in X_{\mathrm{R}\geq}$, then $unch(x) \in \mathrm{Ext}(\varphi)$
4. If $x \in X_{\mathrm{U}\geq}$ or $x \in X_{\mathrm{R}\leq}$, then $rst(x) \in \mathrm{Ext}(\varphi)$
5. If $x \sim d \in \mathrm{BF}(\varphi)$, then $\overline{x \sim d} \in \mathrm{Ext}(\varphi)$
6. If $\Phi_1, \Phi_2 \in \mathrm{Ext}(\varphi)$, then $\Phi_1 \wedge \Phi_2, \Phi_1 \vee \Phi_2 \in \mathrm{Ext}(\varphi)$

Extended formulas can be interpreted over extended timed words. An *extended timed word* $\omega = (t_1, a_1, v_1)(t_2, a_2, v_2)(t_3, a_3, v_3)\ldots$ is a sequence where $w = (t_1, a_1), (t_2, a_2)(t_3, a_3)\ldots$ is a timed word over Σ, and for every $i \in \mathbb{N}$, v_i is a clock valuation over $X = X_{U\leq} \cup X_{U\geq} \cup X_{R\leq} \cup X_{R\geq}$ such that for each $x \in X$, either $v_{i+1}(x) = v_i(x) + t_{i+1} - t_i$ or $v_{i+1}(x) = t_{i+1} - t_i$.

The semantics for extended formulas is naturally induced by the semantics of $\mathrm{MTL}_{0,\infty}$ formulas:

Definition 5 Let $\omega = (t_1, a_1, v_1)(t_2, a_2, v_2)(t_3, a_3, v_3)\ldots$ be an extended timed word and $\Phi \in \mathrm{Ext}(\varphi)$. The satisfaction relation $\omega^i \models_e \Phi$ is inductively defined as follows:

1. $\omega^i \models_e x \sim d$ iff $v_i(x) \sim d$, where $\sim \in \{<, \leq, >, \geq\}$.
2. $\omega^i \models_e rst(x)$ iff $v_{i+1}(x) = t_{i+1} - t_i$
3. $\omega^i \models_e unch(x)$ iff $v_{i+1}(x) = v_i(x) + t_{i+1} - t_i$
4. $\omega^i \models_e \phi$ iff $w^i \models \phi$, if $\phi \in \mathrm{Sub}(\varphi)$
5. $\omega^i \models_e \varphi_1 \mathrm{U}_{\sim d-x} \varphi_2$ iff there exists j such that $j \geq i$, $w^j \models \varphi_2$, $t_j - t_i \sim d - v_i(x)$, and $w^k \models \varphi_1$ for all k with $i \leq k < j$, where $\sim \in \{\leq, \geq\}$.
6. $\omega^i \models_e \varphi_1 \mathrm{R}_{\sim d-x} \varphi_2$ iff for all $j \geq i$ such that $t_j - t_i \sim d - v_i(x)$, either $w^j \models \varphi_2$ or there exists k with $i \leq k < j$ and $w^k \models \varphi_1$, where $\sim \in \{\leq, \geq\}$.

7. $\omega^i \models_e \Phi_1 \wedge \Phi_2$ iff $\omega^i \models_e \Phi_1$ and $\omega^i \models_e \Phi_2$
8. $\omega^i \models_e \Phi_1 \vee \Phi_2$ iff $\omega^i \models_e \Phi_1$ or $\omega^i \models_e \Phi_2$
9. $\omega^i \models_e \bigcirc\Phi$ iff $\omega^{i+1} \models_e \Phi$

ω^i is a *model* of Φ if $\omega^i \models_e \Phi$. Two extended formulas are said to be *equivalent* if they have exactly the same models.

4.2 Constructing non-deterministic automata

Let us show how we construct a timed automaton $\mathbb{A}_\varphi = (L, l_0, X, \mathbb{E})$ for φ. L is defined to be $\{\varphi\} \cup 2^{\mathrm{BF}(\varphi)}$, l_0 is φ, and X is $X_{\mathrm{U}\leq} \cup X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq} \cup X_{\mathrm{R}\geq}$.

For each $\psi \in L$, we will define how to compute the set of outgoing transitions from ψ. The core of the computation is a rewriting function β that tells what formula should be satisfied in the next observation and which clocks should be reset when we see this observation. β is defined inductively for all formulas in $\mathrm{Sub}(\varphi) \cup \mathrm{CL}(\varphi)$, as follows.

1. $\beta(\varphi_1 \,\mathrm{U}\, \varphi_2) = \beta(\varphi_2) \vee (\beta(\varphi_1) \wedge \bigcirc(\varphi_1 \,\mathrm{U}\, \varphi_2))$
2. $\beta(\varphi_1 \,\mathrm{U}_{\leq d}\, \varphi_2) = \beta(\varphi_2) \vee (\beta(\varphi_1) \wedge \bigcirc((x \leq d) \wedge (\varphi_1 \,\mathrm{U}_{\leq d-x}\, \varphi_2)))$, where x is the clock assigned to $\varphi_1 \,\mathrm{U}_{\leq d}\, \varphi_2$
3. $\beta(\varphi_1 \,\mathrm{U}_{\leq d-x}\, \varphi_2) = \beta(\varphi_2) \vee (\beta(\varphi_1) \wedge unch(x) \wedge \bigcirc((x \leq d) \wedge (\varphi_1 \,U_{\leq d-x}\, \varphi_2)))$
4. $\beta(\varphi_1 \,\mathrm{U}_{\geq d}\, \varphi_2) = \beta(\varphi_1) \wedge rst(x) \wedge \bigcirc((\varphi_1 \,\mathrm{U}_{\geq d-x}\, \varphi_2) \vee ((x \geq d) \wedge (\varphi_1 \,\mathrm{U}\, \varphi_2)))$, where x is the clock assigned to $\varphi_1 \,\mathrm{U}_{\geq d}\, \varphi_2$
5. $\beta(\varphi_1 \,\mathrm{U}_{\geq d-x}\, \varphi_2) = \beta(\varphi_1) \wedge \bigcirc((\varphi_1 \mathrm{U}_{\geq d-x}\, \varphi_2) \vee ((x \geq d) \wedge (\varphi_1 \mathrm{U}\, \varphi_2)))$
6. $\beta(\varphi_1 \,\mathrm{R}\, \varphi_2) = \beta(\varphi_2) \wedge (\beta(\varphi_1) \vee \bigcirc(\varphi_1 \,\mathrm{R}\, \varphi_2))$
7. $\beta(\varphi_1 \,\mathrm{R}_{\leq d}\, \varphi_2) = \beta(\varphi_2) \wedge (\beta(\varphi_1) \vee \mathrm{rst}(x) \wedge \bigcirc((\varphi_1 \,\mathrm{R}_{\leq d-x}\, \varphi_2) \vee (x > d)))$, where x is the clock assigned to $\varphi_1 \,\mathrm{R}_{\leq d}\, \varphi_2$
8. $\beta(\varphi_1 \,\mathrm{R}_{\leq d-x}\, \varphi_2) = \beta(\varphi_2) \wedge (\beta(\varphi_1) \vee \bigcirc((\varphi_1 \,\mathrm{R}_{\leq d-x}\, \varphi_2) \vee (x > d)))$
9. $\beta(\varphi_1 \,\mathrm{R}_{\geq d}\, \varphi_2) = \beta(\varphi_1) \vee \bigcirc(((x < d) \wedge (\varphi_1 \,\mathrm{R}_{\geq d-x}\, \varphi_2)) \vee (\varphi_1 \,\mathrm{R}\, \varphi_2))$, where x is the clock assigned to $\varphi_1 \mathrm{R}_{\geq d}\, \varphi_2$
10. $\beta(\varphi_1 \,\mathrm{R}_{\geq d-x}\, \varphi_2) = \beta(\varphi_1) \vee (unch(x) \wedge \bigcirc(((x < d) \wedge (\varphi_1 \,\mathrm{R}_{\geq d-x}\, \varphi_2)) \vee (\varphi_1 \,\mathrm{R}\, \varphi_2)))$
11. $\beta(\varphi_1 \wedge \varphi_2) = \beta(\varphi_1) \wedge \beta(\varphi_2)$
12. $\beta(\varphi_1 \vee \varphi_2) = \beta(\varphi_1) \vee \beta(\varphi_2)$
13. $\beta(\bigcirc\varphi_1) = \bigcirc(\varphi_1)$
14. $\beta(\varphi_1) = \varphi_1$, if φ_1 is an action or a clock bound
15. $\beta(true) = true$
16. $\beta(false) = false$.

It is obvious that $\beta(\psi)$ is an extended formula in $\mathrm{Ext}(\varphi)$.

From the semantics given in Sect. 2 for $\mathrm{MTL}_{0,\infty}$, we know that $(\bigvee_{a\in\Sigma} a) \equiv true$ and for any $a, b \in \Sigma$, if $a \neq b$, then $a \wedge b \equiv false$. Using these facts and that $\bigcirc$ distributes over disjunction and conjunction, we can show by induction that $\beta(\psi)$ can be transformed equivalently into a disjunction of the following form:

$$\bigvee_{j=1}^{k} \left(a_j \wedge g_j \wedge rst(X_j) \wedge unch(Y_j) \wedge \bigcirc(\psi_j)\right)$$

where for every j between 1 and k: $a_j \in \Sigma$ is an action, g_j is a conjunction of clock bounds, $X_j \subseteq (X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq})$ and $Y_j \subseteq (X_{\mathrm{U}\leq} \cup X_{\mathrm{R}\geq})$, $\psi_j \in 2^{\mathrm{BF}(\varphi)}$, $rst(X_j)$ is the abbreviation of $\bigwedge_{x\in X_j} rst(x)$ and $unch(Y_j)$ is the abbreviation of $\bigwedge_{x\in Y_j} unch(x)$.

We call each $a_j \wedge g_j \wedge rst(X_j) \wedge unch(Y_j) \wedge \bigcirc(\psi_j)$ a basic conjunction of $\beta(\psi)$.

From each basic conjunction $a_j \wedge g_j \wedge rst(X_j) \wedge unch(Y_j) \wedge \bigcirc(\psi_j)$, we can define a group of transitions from ψ to ψ_j:

$$(\psi, a_j, g_j, r, \psi_j) \in \mathbb{E} \quad iff\ X_j \subseteq r \subseteq (X \backslash Y_j)$$

Theorem 1 *Let φ be a safety $MTL_{0,\infty}$ formula over Σ, and $\mathbb{A}_\varphi$ be a TA for φ built according to the procedure given above. Then $\mathcal{L}(\mathbb{A}_\varphi) \backslash Zeno(\Sigma) = \mathcal{L}(\varphi) \backslash Zeno(\Sigma)$.*

Theorem 2 *Let φ be a co-safety $MTL_{0,\infty}$ formula over Σ, and $\mathbb{A}_\varphi$ be a TA for φ built according to the procedure given above. Then $\mathcal{L}_{reach}(\mathbb{A}_\varphi, \{true\}) \backslash Zeno(\Sigma) = \mathcal{L}(\varphi) \backslash Zeno(\Sigma)$, where $\mathcal{L}_{reach}(\mathbb{A}_\varphi, \{true\})$ denotes the timed words that are accepted by a run that eventually reaches the location $true$(that is, the empty subset of $BF(\varphi)$).*

4.3 Proofs for Theorem 1 and Theorem 2

In this subsection we will provide proofs for Theorem 1 and Theorem 2. The following two lemmas proves the validity of the rewrite rules.

Lemma 1 *Let ω be an extended timed word and $\psi \in Sub(\varphi) \cup CL(\varphi)$. If $\omega \models_e \beta(\psi)$, then $\omega \models_e \psi$.*

Proof We prove this by induction on the structure of ψ.

1. If ψ is an action or a clock bound, then $\beta(\psi) = \psi$ and the conclusion is true.
2. Assume that the conclusion is true for all sub-formulas of ψ.

Case 1. $\psi = \psi_1 \wedge \psi_2$:
Since $\beta(\psi) = \beta(\psi_1) \wedge \beta(\psi_2)$, it is easy to see that the conclusion is true for ψ.
Case 2. $\psi = \varphi_1 U_{\leq d}\, \varphi_2$:

1. If $\omega \models_e \beta(\varphi_2)$, then $\omega \models_e \varphi_2$, and so $\omega \models_e \varphi_1 U_{\leq d}\, \varphi_2$.
2. If $\omega \not\models_e \beta(\varphi_2)$, then from $\omega \models_e \beta(\psi)$ and $\beta(\psi) = \beta(\varphi_2) \vee (\beta(\varphi_1) \wedge \bigcirc((x \leq d) \wedge (\varphi_1\, \mathrm{U}_{\leq d-x}\, \varphi_2)))$, we know that $\omega \models_e \beta(\varphi_1) \wedge \bigcirc((x \leq d) \wedge (\varphi_1\, \mathrm{U}_{\leq d-x}\, \varphi_2))$. Hence $\omega \models_e \beta(\varphi_1)$ and $\omega \models_e \bigcirc((x \leq d) \wedge (\varphi_1\, \mathrm{U}_{\leq d-x}\, \varphi_2))$. From the induction assumption we get that $\omega \models_e \varphi_1$. From $\omega \models_e \varphi_1$ and $\omega \models_e \bigcirc((x \leq d) \wedge (\varphi_1\, \mathrm{U}_{\leq d-x}\, \varphi_2)))$ we then get the conclusion that $\omega \models_e \varphi_1 U_{\leq d}\, \varphi_2$.

The proofs for the rest of the cases are similar and are omitted here. □

Definition 6 Given a timed word $w = (t_1, a_1), (t_2, a_2)(t_3, a_3) \ldots$ and a clock valuation $v_1 = \overline{0}$, an extended timed word $\overline{w} = (t_1, a_1, v_1)(t_2, a_2, v_2)(t_3, a_3, v_3) \ldots$ can be defined inductively as follows:

1. If x is the clock assigned to $\varphi_1 \mathrm{U}_{\leq d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$, then

$$v_{i+1}(x) = \begin{cases} v_i(x) + t_{i+1} - t_i, & \text{if } v_i(x) \leq d,\ w^i \models \varphi_1 \mathrm{U}_{\leq d - v_i(x)}\, \varphi_2 \text{ and } w^i \not\models \varphi_2; \\ t_{i+1} - t_i, & \text{otherwise.} \end{cases}$$

2. If x is the clock assigned to $\varphi_1 \mathrm{U}_{\geq d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$, then

$$v_{i+1}(x) = \begin{cases} t_{i+1} - t_i, & \text{if } w^i \models \varphi_1 \mathrm{U}_{\geq d}\, \varphi_2; \\ v_i(x) + t_{i+1} - t_i, & \text{otherwise.} \end{cases}$$

3. If x is the clock assigned to $\varphi_1 \mathrm{R}_{\leq d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$, then

$$v_{i+1}(x) = \begin{cases} t_{i+1} - t_i, & \text{if} w^i \models \varphi_1 \mathrm{R}_{\leq d}\, \varphi_2 \text{ and } w^i \not\models \varphi_1; \\ v_i(x) + t_{i+1} - t_i, & \text{otherwise.} \end{cases}$$

4. If x is the clock assigned to $\varphi_1 \mathrm{R}_{\geq d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$, then

$$v_{i+1}(x) = \begin{cases} v_i(x) + t_{i+1} - t_i, & \text{if } v_i(x) < d,\ w^i \models \varphi_1 \mathrm{R}_{\geq d - v_i(x)}\, \varphi_2 \text{ and } w^i \not\models \varphi_1; \\ t_{i+1} - t_i, & \text{otherwise.} \end{cases}$$

Lemma 2 *Let w be a timed word, and $\overline{w}$ be the extended timed word defined in Definition 6, then for every $\psi \in Sub(\varphi) \cup CL(\varphi)$, if $\overline{w} \models_e \psi$ then $\overline{w} \models_e \beta(\psi)$.*

Proof By induction on ψ. □

Lemma 3 *For each $\psi \in Sub(\varphi) \cup CL(\varphi)$, $\beta(\psi)$ can be transformed equivalently into a disjunction of basic conjunctions:*

$$\bigvee_{j=1}^{k} \left(a_j \wedge g_j \wedge rst(X_j) \wedge unch(Y_j) \wedge \bigcirc(\psi_j)\right)$$

where for each j between 1 and k : $a_j \in \Sigma$, g_j is a conjunction of clock bounds, $X_j \subseteq (X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq})$, $Y_j \subseteq (X_{\mathrm{U}\leq} \cup X_{\mathrm{R}\geq})$, and $\psi_j \in 2^{\mathrm{BF}(\varphi)}$.

Proof We first define Length(ψ) for each $\psi \in Sub(\varphi) \cup CL(\varphi)$ as follows.

1. Length$(true) = 0$;
2. Length$(false) = 0$;
3. Length$(\psi_1) = 0$, if ψ_1 is an action or a clock bound;
4. Length$(\bigcirc\psi_1) = 0$;
5. Length$(\psi_1 op \psi_2)$ = Length(ψ_1)+ Length$(\psi_2) + 1$, where op is an operator in the set $\{\wedge, \vee, \mathrm{U}, \mathrm{U}_{\leq d}, \mathrm{U}_{\geq d}, \mathrm{U}_{\leq d-x}, \mathrm{U}_{\geq d-x}, \mathrm{R}, \mathrm{R}_{\leq d}, \mathrm{R}_{\geq d}, \mathrm{R}_{\leq d-x}, \mathrm{R}_{\geq d-x}\}$.

Now we prove the conclusion by induction on Length(ψ).

1. Since $\beta(true) = true \equiv \bigvee_{a\in\Sigma} a \equiv \bigvee_{a\in\Sigma} \left(a \wedge \bigcirc(true)\right)$, the conclusion is true for the case of $\psi = true$.
2. Since $\beta(false) = false$ is the disjunction of zero disjuncts, the conclusion is also true for the case of $\psi = false$.
3. If $\psi = a$, and a is an action, then $\beta(\psi) = a \equiv (a \wedge \bigcirc(true))$, and the conclusion is true for the case of $\psi = a$.
4. If ψ is a clock bound $x \sim d$, then $\beta(\psi) = (x \sim d) \equiv \bigvee_{a\in\Sigma} \left(a \wedge (x \sim d) \wedge \bigcirc(true)\right)$. So the conclusion is true for $\psi = (x \sim d)$.
5. If $\psi = \bigcirc\psi_1$, then $\beta(\psi) = \bigcirc(\psi_1) \equiv \bigvee_{a\in\Sigma} \left(a \wedge \bigcirc(\psi_1)\right)$. So the conclusion is true for this case.
6. If $\psi = \varphi_1 \wedge \varphi_2$, then $\beta(\psi) = \beta(\varphi_1) \wedge \beta(\varphi_2)$. By the induction assumption, we have that

$$\beta(\varphi_1) \equiv \bigvee_{i=1}^{k_1} \left(a'_i \wedge g'_i \wedge rst(X'_i) \wedge unch(Y'_i) \wedge \bigcirc(\psi'_i)\right)$$

and

$$\beta(\varphi_2) \equiv \bigvee_{j=1}^{k_2} \left(a''_j \wedge g''_j \wedge rst(X''_j) \wedge unch(Y''_j) \wedge \bigcirc(\psi''_j)\right),$$

then

$$\beta(\psi) \equiv \bigvee_{i=1}^{k_1} \bigvee_{j=1}^{k_2} \left((a'_i \wedge a''_j) \wedge (g'_i \wedge g''_j) \wedge rst(X'_i \cup X''_j) \wedge unch(Y'_i \cup Y''_j) \wedge \bigcirc(\psi'_i \wedge \psi''_j)\right)$$

Since $a'_i \wedge a''_j$ is either equals to a'_i or $false$, it can be concluded that the conclusion is true for the case of $\psi = \varphi_1 \wedge \varphi_2$.

7. The other cases can be proved similarly. □

Now let us begin to prove the Theorem 1.

Proof of Theorem 1 1. $\mathcal{L}(\mathbb{A}_\varphi) \backslash Zeno(\Sigma) \subseteq \mathcal{L}(\varphi) \backslash Zeno(\Sigma)$.

Let $w = (t_1, a_1), (t_2, a_2), (t_3, a_3) \ldots$ be a non-Zeno timed word in $\mathcal{L}(\mathbb{A}_\varphi)$, then there exist $\psi_1, \psi_2, \psi_3, \ldots \in L$, $(\psi_1, a_1, g_1, r_1, \psi_2), (\psi_2, a_2, g_2, r_2, \psi_3), \ldots \in \mathbb{E}$ and $v_1, v_2, v_3, \ldots \in \mathbb{R}^X_{\geq 0}$ such that $\psi_1 = \varphi$, and for each $i \geq 1$: there are $X_i \subseteq (X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq})$ and $Y_i \subseteq (X_{\mathrm{U}\leq} \cup X_{\mathrm{R}\geq})$ such that $a_i \wedge g_i \wedge rst(X_i) \wedge unch(Y_i) \wedge \bigcirc(\psi_{i+1})$ is a basic conjunction of $\beta(\psi_i)$, $X_i \subseteq r_i \subseteq (X \backslash Y_i)$, $v_i \models g_i$, and $v_{i+1} = (v_i[r_i]) + (t_{i+1} - t_i)$.

Then we get an extended timed word $\omega = (t_1, a_1, v_1), (t_2, a_2, v_2), (t_3, a_3, v_3), \ldots$, and $\omega^i \models_e a_i \wedge g_i \wedge rst(X_i) \wedge unch(Y_i)$.

For each $i \geq 1$, using ω^i and ψ_{i+1}, we can define an assignment $\mu_i(\psi) \in \{true, false\}$ for all extended formulas in Ext(φ) as follows.

1. For each $a \in \Sigma$, $\mu_i(a) = true$ iff $a = a_i$
2. $\mu_i(x \sim d) = true$ iff $v_i(x) \sim d$
3. $\mu_i(rst(x)) = true$ iff $v_{i+1}(x) = t_{i+1} - t_i$.
4. $\mu_i(unch(x)) = true$ iff $v_{i+1}(x) = v_i(x) + t_{i+1} - t_i$
5. For each $\varphi_1 \in \mathrm{BF}(\varphi)$, $\mu_i(\bigcirc(\varphi_1)) = true$ iff $\varphi_1 \in \psi_{i+1}$ (please noted that ψ_{i+1} is a subset of BF(φ))
6. $\mu_i(\varphi_1 \wedge \varphi_2) = \mu_i(\varphi_1) \wedge \mu_i(\varphi_2)$
7. $\mu_i(\varphi_1 \vee \varphi_2) = \mu_i(\varphi_1) \vee \mu_i(\varphi_2)$
8. $\mu_i(\varphi_1 \,\mathrm{R}\, \varphi_2) = \mu_i(\varphi_2) \wedge (\mu_i(\varphi_1) \vee \mu_i(\bigcirc(\varphi_1 \,\mathrm{R}\, \varphi_2)))$
9. $\mu_i(\varphi_1 \,\mathrm{R}_{\leq d}\, \varphi_2) = \mu_i(\varphi_2) \wedge (\mu_i(\varphi_1) \vee \mu_i(\mathrm{rst}(x)) \wedge (\mu_i(\bigcirc(\varphi_1 \,\mathrm{R}_{\leq d-x} \varphi_2)) \vee \mu_i(\bigcirc(x > d))))$, where x is the clock assigned to $\varphi_1 \,\mathrm{R}_{\leq d}\, \varphi_2$
10. The other cases for $\mu_i(\varphi_1 \,\mathrm{U}_{\leq d}\, \varphi_2)$, $\mu_i(\varphi_1 \,\mathrm{U}_{\leq d-x}\, \varphi_2)$, $\mu_i(\varphi_1 \,\mathrm{R}_{\leq d-x}\, \varphi_2)$ can be defined similarly, accordig to the rewriting rules in Sect. 4.2.

It is easy to show that if $a \wedge g \wedge \mathrm{rst}(X_1) \wedge \mathrm{unch}(Y_1) \wedge \bigcirc(\psi')$ is a basic conjunction of $\beta(\psi)$, and $\mu_i(a \wedge g \wedge \mathrm{rst}(X_1) \wedge \mathrm{unch}(Y_1) \wedge \bigcirc(\psi'))$ is *true*, then $\mu_i(\psi)$ is *true*.

Thus for each $i \geq 1$, we get that $\mu_i(\psi_i) = true$, and furthermore, for each basic formula $\psi \in \psi_i$, we have $\mu_i(\psi) = true$.

Now we show that for each $\psi \in \mathrm{Sub}(\varphi) \cup \mathrm{CL}(\varphi)$, if $\mu_i(\psi) = true$, then $\omega^i \models_e \psi$.

1. If ψ is an action or a clock bound, the conclusion is obviously true.
2. If $\psi = \bigcirc(\phi)$, and $\mu_i(\psi) = true$,
   then $\phi \in \psi_{i+1}$, and $\mu_{i+1}(\phi) = true$.
   By induction, we get that $\omega^{i+1} \models_e \phi$.
   So $\omega^i \models_e \bigcirc(\phi)$.
3. If $\psi = \phi_1 \wedge \phi_2$ and $\mu_i(\psi) = true$,
   then $\mu_i(\phi_1) = \mu_i(\phi_2) = true$.
   By induction, we get that $\omega^i \models_e \phi_1$ and $\omega^i \models_e \phi_2$.
   Thus we get the conclusion that $\omega^i \models_e \phi_1 \wedge \phi_2$.

4. If $\psi = \phi_1 \vee \phi_2$ and $\mu_i(\psi) = true$,
   then $\mu_i(\phi_1) = true$ or $\mu_i(\phi_2) = true$.
   By induction, we get that $\omega^i \models_e \phi_1$ or $\omega^i \models_e \phi_2$.
   Thus $\omega^i \models_e \phi_1 \vee \phi_2$.
5. If $\psi = \phi_1 \mathrm{R} \phi_2$ and $\mu_i(\psi) = true$,
   then $\mu_i(\varphi_2) \wedge (\mu_i(\varphi_1) \vee \mu_i(\bigcirc(\varphi_1 \mathrm{R} \varphi_2)))$ is true.

Thus we get that $\mu_i(\varphi_2) = true$, and that $\mu_i(\varphi_1) = true$ or $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+1}$.
By induction, we get that $\omega^i \models_e \varphi_2$, $\omega^i \models_e \varphi_1$ or $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+1}$.

Case 1. If $\omega^i \models_e \varphi_2$ and $\omega^i \models_e \varphi_1$, then $\omega^i \models_e \phi_1 \mathrm{R} \phi_2$, and the conclusion is true.
Case 2. If $\omega^i \models_e \varphi_2$ and $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+1}$, then from $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+1}$ we know that $\mu_{i+1}(\varphi_1 \mathrm{R} \varphi_2) = true$, thus $\mu_{i+1}(\varphi_2) = true$, and $\mu_{i+1}(\varphi_1) = true$ or $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+2}$.

By induction, we have that $\omega^{i+1} \models_e \varphi_2$, $\omega^{i+1} \models_e \varphi_1$ or $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+2}$.

(2.1) If $\omega^{i+1} \models_e \varphi_2$ and $\omega^{i+1} \models_e \varphi_1$, then from the fact that $\omega^i \models_e \varphi_2$, we know that $\omega^i \models_e \phi_1 \mathrm{R} \phi_2$. So the conclusion is true.
(2.2) If $\omega^{i+1} \models_e \varphi_2$ and $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+2}$, then we know that $\mu_{i+2}(\varphi_1 \mathrm{R} \varphi_2) = true$.

Thus $\mu_{i+2}(\varphi_2) = true$, and $\mu_{i+2}(\varphi_1) = true$ or $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+3}$.
By induction, we get that $\omega^{i+2} \models_e \varphi_2$, $\omega^{i+2} \models_e \varphi_1$ or $\varphi_1 \mathrm{R} \varphi_2 \in \psi_{i+3}$.
This procedure can stop or proceed infinitely; in both case, we could get that $\omega^i \models_e \phi_1 \mathrm{R} \phi_2$.

6. We omit the proofs for the other cases.

Since $\mu_1(\varphi) = true$, from the above conclusion we get that $\omega^1 \models_e \varphi$, thus we finish the proof for $\mathcal{L}(\mathbb{A}_\varphi) \backslash Zeno(\Sigma) \subseteq \mathcal{L}(\varphi) \backslash Zeno(\Sigma)$.

2. $\mathcal{L}(\varphi) \backslash Zeno(\Sigma) \subseteq \mathcal{L}(\mathbb{A}_\varphi) \backslash Zeno(\Sigma)$.

Let $w = (t_1, a_1), (t_2, a_2), (t_3, a_3) \ldots$ be a non-Zeno timed word in $\mathcal{L}(\varphi)$ and $\overline{w}$=$(t_1, a_1, v_1)(t_2, a_2, v_2)(t_3, a_3, v_3) \ldots$ be the extended timed word defined in Definition 6.

From $w \in \mathcal{L}(\varphi)$, we know that $\overline{w} \models_e \varphi$.

Let $\varphi_1 = \varphi$, then by Lemma 2, we get that $\overline{w} \models_e \beta(\varphi_1)$. Since $\beta(\varphi_1)$ can be written as a disjunction of some basic conjunctions, so there is a basic conjunction $\alpha_1 \wedge g_1 \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\varphi_2)$ of φ_1 such that $\overline{w} \models_e \alpha_1 \wedge g_1 \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\varphi_2)$.

Thus $\alpha_1 = a_1$, $v_1 \models g_1$, and $\overline{w} \models_e rst(X_1) \wedge unch(Y_1)$, and $\overline{w^2} \models_e \varphi_2$.

Let r_1={$x \mid x \in X \backslash Y_1, and v_2(x) = t_2 - t_1$}, then $X_1 \subseteq r_1 \subseteq (X \backslash Y_1)$.

So from the construction in Sect. 4.2, $(\varphi_1, a_1, g_1, r_1, \varphi_2) \in \mathbb{E}$ is a transition of $\mathbb{A}_\varphi$, $\overline{w}^2 \models \varphi_2$, and $v_2 = v_1[r_1] + (t_2 - t_1)$.

Similarly, from $\overline{w}^2 \models_e \varphi_2$, by Lemma 2, we know that there is a basic conjunction $\alpha_2 \wedge g_2 \wedge rst(X_2) \wedge unch(Y_2) \wedge \bigcirc(\varphi_3)$ of φ_2 such that $\overline{w^2} \models_e \alpha_2 \wedge g_2 \wedge rst(X_2) \wedge unch(Y_2) \wedge \bigcirc(\varphi_3)$.

Thus $\alpha_2 = a_2$, $v_2 \models g_2$, and $\overline{w^2} \models_e rst(X_2) \wedge unch(Y_2)$, and $\overline{w^3} \models_e \varphi_3$.

Let r_2={$x \mid x \in X \backslash Y_2$, and $v_3(x) = t_3 - t_2$}, then $X_2 \subseteq r_2 \subseteq (X \backslash Y_2)$.

Thus $(\varphi_2, a_2, g_2, r_2, \varphi_3) \in \mathbb{E}$ is a transition of $\mathbb{A}_\varphi$, $\overline{w}^3 \models_e \varphi_3$, and $v_3 = v_2[r_2] + (t_3 - t_2)$.

By repeating above reasoning, we can get a run of $\mathbb{A}_\varphi$ that accepts $w = (t_1, a_1), (t_2, a_2), (t_3, a_3) \ldots$. This completes our proof for Theorem 1. □

Now we turn to the proof for Theorem 2.

Proof of Theorem 2 1. $\mathcal{L}_{reach}(\mathbb{A}_\varphi, \{true\}) \backslash Zeno(\Sigma) \subseteq \mathcal{L}(\varphi) \backslash Zeno(\Sigma)$

Let $w = (t_1, a_1), (t_2, a_2), (t_3, a_3) \ldots$ be a non-Zeno timed word in $\mathcal{L}_{reach}(\mathbb{A}_\varphi, \{true\})$, then there are $\psi_1, \psi_2, \psi_3, \ldots \in L$ and $v_1, v_2, v_3, \ldots \in \mathbb{R}^X_{\geq 0}$ such that $\psi_1 =$

$\varphi, (\psi_i, a_i, g_i, r_i, \psi_{i+1}) \in \mathbb{E}$ is a transition of $\mathbb{A}_\varphi$, and $a_i \wedge g_i \wedge rst(X_i) \wedge unch(Y_i) \wedge \bigcirc(\psi_{i+1})$ is a basic conjunction of $\beta(\psi_i)$, $X_i \subseteq r_i \subseteq (X \backslash Y_i)$, $v_i \models g_i$, and $v_{i+1} = (v_i[r_i]) + (t_{i+1} - t_i)$.

Then we get an extended timed word $\omega = (t_1, a_1, v_1), (t_2, a_2, v_2), (t_3, a_3, v_3), \ldots$, and $\omega^i \models_e a_i \wedge g_i \wedge rst(X_i) \wedge unch(Y_i)$.

Because $\psi_1, \psi_2, \psi_3, \ldots$ could reach the location $true$, we can assume that $\psi_n = true$.

Now we prove by induction that for all $i \leq n$: $\omega^i \models_e \psi_i$.

1. If $i = n$, then $\psi_n = true$ and $\omega^n \models_e \psi_n$.
2. Assume $\omega^i \models_e \psi_i$ is true for all $i \geq k+1$, now we show that $\omega^k \models_e \psi_k$ is true.
   From $(\psi_k, a_k, g_k, r_k, \psi_{k+1}) \in \mathbb{E}$, we know that $\omega^k \models_e a_k \wedge g_k \wedge rst(X_k) \wedge unch(Y_k) \wedge \bigcirc(\psi_{k+1})$, so $\omega^k \models_e \beta(\psi_k)$.

Then from Lemma 1, we get the conclusion that $\omega^k \models_e \psi_k$.

2. $\mathcal{L}(\varphi) \backslash Zeno(\Sigma) \subseteq \mathcal{L}_{reach}(\mathbb{A}_\varphi, \{true\}) \backslash Zeno(\Sigma)$.

Let $w = (t_1, a_1), (t_2, a_2), (t_3, a_3) \ldots$ be a non-Zeno timed word in $\mathcal{L}(\varphi)$ and $\overline{w}$=$(t_1, a_1, v_1)(t_2, a_2, v_2)(t_3, a_3, v_3) \ldots$ be the extended timed word defined in Definition 6.

From $w \in \mathcal{L}(\varphi)$, we know that $\overline{w} \models_e \varphi$.

Let $\varphi_1 = \varphi$, there is a basic conjunction $\alpha_1 \wedge g_1 \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\varphi_2)$ of φ_1 such that $\overline{w} \models \alpha_1 \wedge g_1 \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\varphi_2)$.

Then $\alpha_1 = a_1$, $v_1 \models g_1$, and $\overline{w} \models rst(X_1) \wedge unch(Y_1)$, and $\overline{w}^2 \models \varphi_2$.

Let r_1=$\{x \mid x \in (X \backslash Y_1), and v_2(x) = t_2 - t_1\}$, then $X_1 \subseteq r_1 \subseteq (X \backslash Y_1)$.

So from the construction in Section 3.2, $(\varphi_1, a_1, g_1, r_1, \varphi_2) \in \mathbb{E}$ is a transition of $\mathbb{A}_\varphi$, $\overline{w}^2 \models \varphi_2$, and $v_2 = (v_1[r_1]) + (t_2 - t_1)$.

Similarly, we can get a sequence $\varphi_2, \varphi_3, \varphi_4, \ldots$ of formulas from $2^{\mathrm{BF}(\varphi)}$ such that $(\varphi_i, a_i, g_i, r_i, \varphi_{i+1}) \in \mathbb{E}$ is a transition of $\mathbb{A}_\varphi$ and $\overline{w}_i \models a_i \wedge g_i \wedge rst(X_i) \wedge unch(Y_i) \wedge \bigcirc(\varphi_{i+1})$ for all $i \in N$.

Now, it suffices to prove that there exists some k such that $\varphi_k = true$.

To do this, we define the depth $\mathrm{dep}(\phi)$ for formulas in $\mathrm{Sub}(\varphi) \cup 2^{\mathrm{BF}(\varphi)}$.

1. $\mathrm{dep}(a) = \mathrm{dep}(x \sim d) = \mathrm{dep}(true) = 0$;
2. $\mathrm{dep}(\phi_1 \vee \phi_2) = \mathrm{dep}(\phi_1 \wedge \phi_2) = \max\{\mathrm{dep}(\phi_1), \mathrm{dep}(\phi_2)\}$;
3. $\mathrm{dep}(\bigcirc\phi) = \mathrm{dep}(\phi_1) + 1$;
4. $\mathrm{dep}(\phi_1 \,\mathrm{U}_{\leq d}\, \phi_2) = \mathrm{dep}(\phi_1 \,\mathrm{R}_{\leq d}\, \phi_2) = \max\{\mathrm{dep}(\phi_1), \mathrm{dep}(\phi_2)\} + 2$;
5. $\mathrm{dep}(\phi_1 \,\mathrm{U}_{\geq d}\, \phi_2) = \max\{\mathrm{dep}(\phi_1), \mathrm{dep}(\phi_2)\} + 3$;
6. $\mathrm{dep}(\phi_1 \,\mathrm{U}_{\geq d-x}\, \phi_2) = \max\{\mathrm{dep}(\phi_1), \mathrm{dep}(\phi_2)\} + 2$;
7. $\mathrm{dep}(\phi_1 \,\mathrm{U}\, \phi_2) = \max\{\mathrm{dep}(\phi_1), \mathrm{dep}(\phi_2)\} + 1$;
8. $\mathrm{dep}(\phi_1 \,\mathrm{U}_{\leq d-x}\, \phi_2) = \mathrm{dep}(\phi_1 \,\mathrm{R}_{\leq d-x}\, \phi_2) = \max\{\mathrm{dep}(\phi_1), \mathrm{dep}(\phi_2)\} + 1$.

Then $\mathrm{dep}(\varphi_1) \geq \mathrm{dep}(\varphi_2) \geq \mathrm{dep}(\varphi_3) \geq \ldots$, and there exists N such that for all $i \geq N$: $\mathrm{dep}(\varphi_i) = \mathrm{dep}(\varphi_N)$.

If $\mathrm{dep}(\varphi_N) > 0$, then some $\phi_1 \,\mathrm{U}_{\leq d-x}\, \phi_2$, $\phi_1 \,\mathrm{R}_{\leq d-x}\, \phi_2$, $\phi_1 \,\mathrm{U}_{\geq d-x}\, \phi_2$ or $\phi_1 \,\mathrm{U}\, \phi_2$ will remain in φ_i for all $i \geq N$.

1. If $\phi_1 \,\mathrm{U}_{\leq d-x}\, \phi_2$, or $\phi_1 \,\mathrm{R}_{\leq d-x}\, \phi_2$, or $\phi_1 \,\mathrm{U}_{\geq d-x}$ is remain in φ_i for all $i \geq N$, then we get that x will not be reset and will not exceed d.
   This is not possible, because the extended timed word $\overline{w}$ is assumed to be non-Zeno.
2. If $\phi_1 \,\mathrm{U}\, \phi_2$ is remain in φ_i for all $i \geq N$, then $\overline{w}^i \models \phi_1 \,\mathrm{U}\, \phi_2$ will be true for all $i \geq N$, and $\overline{w}^i \models \phi_2$ is false for all $i \geq N$. This is also not possible!

Thus $\mathrm{dep}(\varphi_N)$ must be zero, so $\varphi_{N+1} = true$ and $w \in \mathcal{L}_{reach}(\mathbb{A}_\varphi, \{true\}) \backslash Zeno(\Sigma)$. □

4.4 Reducing the transitions in $\mathbb{A}_\varphi$

Given a basic conjunction $a \wedge g \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\psi_1)$, its sub-formula $rst(X_1) \wedge unch(Y_1)$ tells us that the clocks in X_1 should be reset and the clocks in Y_1 should not be reset. It does not tell us the other clocks in $X \backslash (X_1 \cup Y_1)$ should be reset or not be reset. So in the construction of Sect. 4.2 we enumerate all the possible situations for clocks in $X \backslash (X_1 \cup Y_1)$, and thus we get a group of transitions from a basic conjunction. In this subsection, we will show that it is not necessary to enumerate all possible combinations for clocks in $X \backslash (X_1 \cup Y_1)$, there exists a best choice, that is, to reset all clocks in $(X_{\mathrm{U}\leq} \cup X_{\mathrm{R}\geq}) \backslash Y_1$ to zero and keep all clocks in $(X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq}) \backslash X_1$ unchanged. The idea behind this choice is that each clock $x \in (X_{\mathrm{U}\leq} \cup X_{\mathrm{R}\geq})$ should be reset to zero unless $unch(x)$ is asked to be true, and each clock $x \in (X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq})$ should not be reset unless $rst(x)$ is asked to be true. In this way, for a given basic conjunction $a \wedge g \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\psi_1)$ of $\beta(\psi)$, then transition $(\psi, a, g, \lambda, \psi_1)$ with $\lambda = (X_1 \cup ((X_{U\leq} \cup X_{R\geq}) \backslash Y_1))$ will be the unique transition from ψ to ψ_1 in the following construction.

Definition 7 Let φ be an $\mathrm{MTL}_{0,\infty}$ formula over Σ, we can define a timed automaton $\mathcal{A}_\varphi = (L, l_0, X, E)$ for φ as follows.

- $L = \{\varphi\} \cup 2^{\mathrm{BF}(\varphi)}$ is the set of locations, and $l_0 = \varphi$ is the initial location;
- $X = X_{\mathrm{U}\leq} \cup X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq} \cup X_{\mathrm{R}\geq}$ is the set of all clocks.
- $(\psi_1, a, g, \lambda, \psi_2) \in E$ iff there exist $X_1 \subseteq X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq}$ and $Y_1 \subseteq X_{\mathrm{U}\leq} \cup X_{\mathrm{R}\geq}$ such that $a \wedge g \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\psi_2)$ is a basic conjunction of $\beta(\psi_1)$ and $\lambda = (X_1 \cup ((X_{U\leq} \cup X_{R\geq}) \backslash Y_1))$.

Theorem 3 *Let φ be a safety $MTL_{0,\infty}$ formula over Σ, then $\mathcal{L}(\mathcal{A}_\varphi) \backslash Zeno(\Sigma) = \mathcal{L}(\mathbb{A}_\varphi) \backslash Zeno(\Sigma)$.*

Proof It suffices to prove that $\mathcal{L}(\mathbb{A}_\varphi) \backslash Zeno(\Sigma) \subseteq \mathcal{L}(\mathcal{A}_\varphi) \backslash Zeno(\Sigma)$.

Let $w = (t_1, a_1), (t_2, a_2), (t_3, a_3) \ldots$ be a non-Zeno timed word in $\mathcal{L}(\mathbb{A}_\varphi)$, then there exist $\psi_1, \psi_2, \psi_3, \ldots \in L$, $(\psi_1, a_1, g_1, r_1, \psi_2), (\psi_2, a_2, g_2, r_2, \psi_3), \ldots \in \mathbb{E}$ and $v_1, v_2, v_3, \ldots \in \mathbb{R}^X_{\geq 0}$ such that $\psi_1 = \varphi$, and for each $i \geq 1$: there are $X_i \subseteq X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq}$ and $Y_i \subseteq X_{\mathrm{U}\leq} \cup X_{\mathrm{R}\geq}$ such that $a_i \wedge g_i \wedge rst(X_i) \wedge unch(Y_i) \wedge \bigcirc(\psi_{i+1})$ is a basic conjunction of $\beta(\psi_i)$, $X_i \subseteq r_i \subseteq X \backslash Y_i$, $v_i \models g_i$, and $v_{i+1} = (v_i[r_i]) + (t_{i+1} - t_i)$.

For each $i \geq 1$, let $\lambda_i = (X_i \cup ((X_{U\leq} \cup X_{R\geq}) \backslash Y_i))$, then $(\psi_i, a_i, g_i, \lambda_i, \psi_{i+1}) \in E$ is a transition in $\mathcal{A}_\varphi$.

We define $v'_1, v'_2, v'_3, \ldots \in \mathbb{R}^X_{\geq 0}$ inductively as follows:

$v'_1 = v_1$, $v'_2 = v'_1[\lambda_1] + (t_2 - t_1), \ldots, v'_{i+1} = (v'_i[\lambda_i]) + (t_{i+1} - t_i), \ldots$.

Then by induction on i we can prove that the following Assertion 1 is true. □

Assertion 1 For each $i \geq 1$ and $x \in X$, if $x \in X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq}$ then $v_i(x) \leq v'_i(x)$; otherwise, $v_i(x) \geq v'_i(x)$.

From the rewrite rules in Sect. 4.2, it is easy to find that the following Assertion 2 is also true.

Assertion 2 For each $x \in X$, if $x \in X_{\mathrm{U}\geq} \cup X_{\mathrm{R}\leq}$, then $x > d$ and $x \geq d$ will not occur in g_i; otherwise, $x < d$ and $x \leq d$ will not occur in g_i.

From Assertion 1, Assertion 2 and $v_i \models g_i$, we can conclude that $v'_i \models g_i$. Thus $(\psi_1, \overline{0}) \xrightarrow{t_1} (\psi_1, v'_1) \xrightarrow{a_1} (\psi_2, v'_1[\lambda_1]) \xrightarrow{t_2 - t_1} (\psi_2, v'_2) \xrightarrow{a_2} (\psi_3, v'_2[\lambda_2]) \xrightarrow{t_3 - t_2} (\psi_3, v'_3) \ldots$ will be a run of $\mathcal{A}_\varphi$ that accepts the timed word $w = (t_1, a_1), (t_2, a_2), (t_3, a_3) \ldots$. This completes our proof.

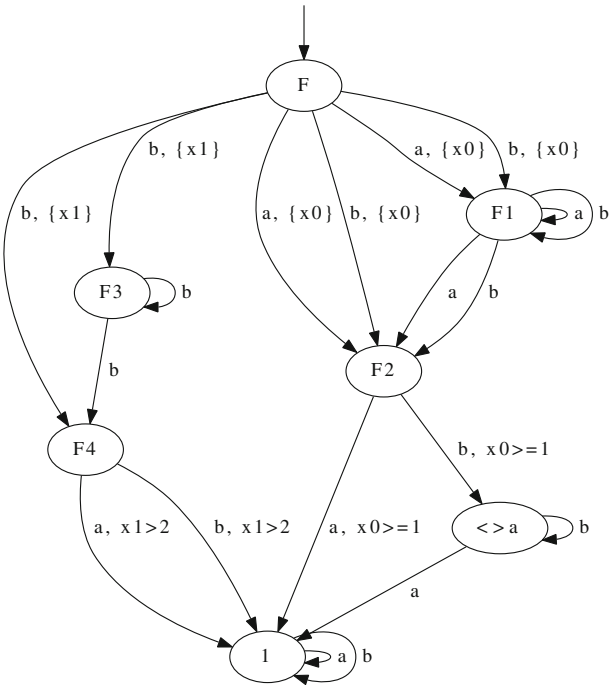

Fig. 2 A timed automaton for $(\Diamond_{\geq 1} a) \vee \Box_{\leq 2} b$

Theorem 4 *Let φ be a co-safety $MTL_{0,\infty}$ formula over Σ, then $\mathcal{L}_{reach}(\mathcal{A}_\varphi, \{true\}) \setminus Zeno(\Sigma) = \mathcal{L}_{reach}(\mathbb{A}_\varphi, \{true\}) \setminus Zeno(\Sigma)$.*

Theorem 4 can be proved similarly as Theorem 3.

Example. Let $\Sigma = \{a, b\}$ and $F = (\Diamond_{\geq 1} a) \vee \Box_{\leq 2} b$, then

$$\begin{aligned}\beta(F) &= (rst(x_0) \wedge \bigcirc(F_1 \vee F_2)) \vee (b \wedge rst(x_1) \wedge \bigcirc(F_3 \vee F_4)) \\ &= (a \wedge rst(x_0) \wedge \bigcirc F_1) \vee (a \wedge rst(x_0) \wedge \bigcirc F_2) \\ &\quad \vee (b \wedge rst(x_0) \wedge \bigcirc F_1) \vee (b \wedge rst(x_0) \wedge \bigcirc F_2) \\ &\quad \vee (b \wedge rst(x_1) \wedge \bigcirc F_3) \vee (b \wedge rst(x_1) \wedge \bigcirc F_4),\end{aligned}$$

where $F_1 = \Diamond_{\geq 1 - x_0} a$, $F_2 = (x_0 \geq 1) \wedge \Diamond a$, $F_3 = (\Box_{\leq 2 - x_1} b)$ and $F_4 = (x_1 > 2)$.

So F has 6 outgoing transitions: $(F, a, true, \{x_0\}, F_1)$, $(F, a, true, \{x_0\}, F_2)$, $(F, b, true, \{x_0\}, F_1)$, $(F, b, true, \{x_0\}, F_2)$, $(F, b, true, \{x_1\}, F_3)$, $(F, b, true, \{x_1\}, F_4)$. Similarly, we can continues compute the outgoing transitions for F_1, F_2, etc. The whole timed automaton constructed for F is presented in Fig. 2.

4.5 Constructing deterministic under-approximation automata

The construction in Definition 7 might produce non-deterministic automata. In fact, as stated earlier, there exist $MTL_{0,\infty}$-formulas for which no equivalent deterministic timed automaton exists. However, if φ is a safety or co-safety MTL $_{0,\infty}$-formula, then we can construct a deterministic under-approximation timed automaton $\mathcal{A}^u_\varphi$ for φ with the construction that we already presented in [12].

The basic idea behind this construction is the classical subset construction from NFA to DFA. The deterministic timed automaton $\mathcal{A}^u_\varphi$ is constructed from non-deterministic automaton $\mathbb{A}_\varphi$ (Sect. 4.2) with the subset-construction technique. Each state in $\mathcal{A}^u_\varphi$ is a subset of the states in $\mathbb{A}_\varphi$, and each transition of $\mathcal{A}^u_\varphi$ can be regarded as a collection of several transitions from $\mathbb{A}_\varphi$.

1. The initial location of $\mathcal{A}^u_\varphi$ is φ, the other locations of $\mathcal{A}^u_\varphi$ are formulas from CL(φ).
2. The actions of $\mathcal{A}^u_\varphi$ are same as that of φ.
3. The clocks of $\mathcal{A}^u_\varphi$ are the set X defined for φ in Sect. 4.1.
4. Let ψ be a formula in $\{\varphi\} \cup \text{CL}(\varphi)$. To define the outgoing transitions from ψ, we further translate $\beta(\psi)$ in disjunctive form into the following deterministic form by repeated use of the logical equivalence $p \Leftrightarrow (p \wedge q) \vee (p \wedge \neg q)$.

$$F_\psi = \bigvee_{i=1}^{n} \left(a_i \wedge g_i \wedge \bigvee_{k=1}^{m_i} (rst(X_{ik}) \wedge unch(Y_{ik}) \wedge \bigcirc(\psi_{ik}))\right)$$

where for all $i \in \{1, \ldots, n\}$: $a_i \in \Sigma$ is an action, g_i is a conjunction of clock bounds, m_i is a positive integer, $X_{ik} \subseteq X_{U\geq} \cup X_{R\leq}$ and $Y_{ik} \subseteq X_{U\leq} \cup X_{R\geq}$ are sets of clocks, $\psi_{ik} \in \text{CL}(\varphi)$, and for all $i \neq j$: $a_i \neq a_j$ or $g_i \wedge g_j$ is false.

Using the facts that $\bigcirc$ distributes over $\vee$, and $rst(Y)$ and $unch(Y)$ are monotonic in Y, the following formula F^u_ψ is an under-approximation of F_ψ:

$$F^u_\psi = \bigvee_{i=1}^{n} \left(a_i \wedge g_i \wedge rst(\bigcup_{k=1}^{m_i} X_{ik}) \wedge unch(\bigcup_{k=1}^{m_i} Y_{ik}) \wedge \bigcirc(\bigvee_{k=1}^{m_i} \psi_{ik})\right)$$

Then ψ has n outgoing transitions in $\mathcal{A}^u_\varphi$, that is, $\{(\psi, a_i, g_i, (\bigcup_{k=1}^{m_i} X_{ik}) \cup ((X_{U\leq} \cup X_{R\geq}) \backslash (\bigcup_{k=1}^{m_i} Y_{ik})), \bigvee_{k=1}^{m_i} \psi_{ik}) \mid 1 \leq i \leq n\}$. We use Tran($\mathcal{A}^u_\varphi$) to denote the set of all transitions in $\mathcal{A}^u_\varphi$.

Please noted that each $a_i \wedge g_i \wedge rst(X_{ik}) \wedge unch(Y_{ik}) \wedge \bigcirc(\psi_{ik})$ is a basic conjunction of $\beta(\psi)$, so each transition in $\mathcal{A}^u_\varphi$ can be considered as a collection of several transitions from the automaton $\mathbb{A}_\varphi$ (Sect. 4.2). Now we show that every timed word accepted by $\mathcal{A}^u_\varphi$ is also accepted by $\mathbb{A}_\varphi$.

Theorem 5 *Let φ be a safety $MTL_{0,\infty}$ formula over Σ, then $\mathcal{L}(\mathcal{A}^u_\varphi) \backslash Zeno(\Sigma) \subseteq \mathcal{L}(\mathbb{A}_\varphi) \backslash Zeno(\Sigma)$.*

Proof Let $w = (t_1, a_1), (t_2, a_2), (t_3, a_3) \ldots$ be a non-Zeno timed word in $\mathcal{L}(\mathcal{A}^u_\varphi)$, then there exist $\psi_1, \psi_2, \psi_3, \ldots \in \{\varphi\} \cup \text{CL}(\varphi)$, $(\psi_1, a_1, g_1, r_1, \psi_2), (\psi_2, a_2, g_2, r_2, \psi_3), \ldots \in \text{Tran}(\mathcal{A}^u_\varphi)$, and $v_1, v_2, v_3, \ldots \in \mathbb{R}^X_{\geq 0}$ such that $\psi_1 = \varphi$, and for each $i \geq 1$: $v_i \models g_i$, and $v_{i+1} = (v_i[r_i]) + (t_{i+1} - t_i)$.

For each $i \geq 1$, since $(\psi_i, a_i, g_i, r_i, \psi_{i+1})$ is a transition of $\mathcal{A}^u_\varphi$, there exist $X_{i1}, X_{i2}, \ldots, X_{im_i} \in 2^{X_{U\geq} \cup X_{R\leq}}$, $Y_{i1}, Y_{i2}, \ldots, Y_{im_i} \in 2^{X_{U\leq} \cup X_{R\geq}}$, and $\psi_{i1}, \psi_{i2}, \ldots, \psi_{im_i} \in 2^{\text{BF}(\varphi)}$ such that

$r_i = (\bigcup_{k=1}^{m_i} X_{ik}) \cup ((X_{U\leq} \cup X_{R\geq}) \backslash (\bigcup_{k=1}^{m_i} Y_{ik}))$, $\psi_{i+1} = \bigvee_{k=1}^{m_i} \psi_{ik}$, and for every $k \in \{1, 2, \ldots, m_i\}$: $a_i \wedge g_i \wedge rst(X_{ik}) \wedge unch(Y_{ik}) \wedge \bigcirc(\psi_{ik})$ is a basic conjunction of $\beta(\psi_i)$.

Now we can defined an infinite tree $\mathcal{T}$ using the formulas from $\{\varphi\} \cup 2^{\text{BF}(\varphi)}$ as follows.

1. $(0, \varphi)$ is the root of $\mathcal{T}$.
2. For each $i \geq 1$ and $k \in \{1, 2, \ldots, m_i\}$, (i, ψ_{ik}) is a node at level i.
3. For each $i \geq 2$ and $k \in \{1, 2, \ldots, m_i\}$, since $a_i \wedge g_i \wedge rst(X_{ik}) \wedge unch(Y_{ik}) \wedge \bigcirc(\psi_{ik})$ is a basic conjunction of $\beta(\psi_i)$ and $\psi_i = \bigvee_{k=1}^{m_{(i-1)}} \psi_{(i-1)k}$, then there exists a n_i such that

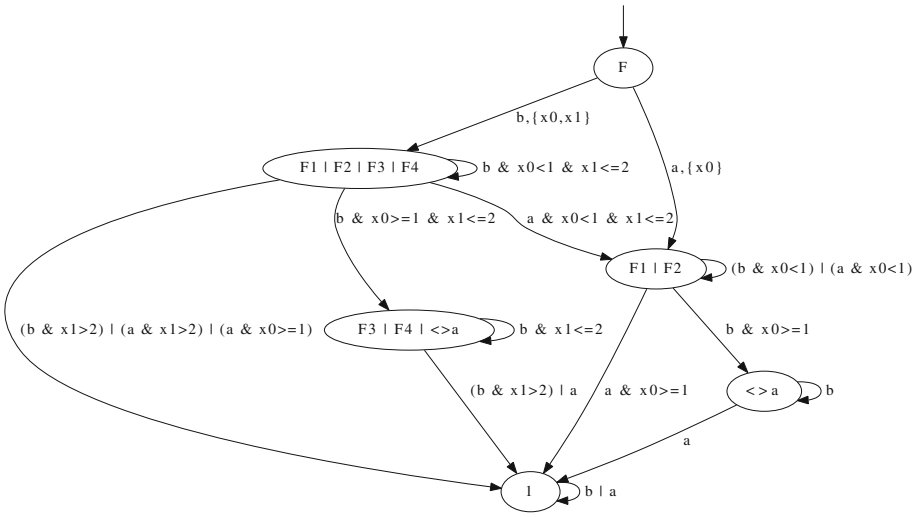

Fig. 3 Under-approximation automaton for $(\Diamond_{\geq 1} a) \vee \Box_{\leq 2} b$

$1 \leq n_i \leq m_{(i-1)}$ and $a_i \wedge g_i \wedge rst(X_{ik}) \wedge unch(Y_{ik}) \wedge \bigcirc(\psi_{ik})$ is a basic conjunction of $\beta(\psi_{(i-1)n_i})$. We define $(i-1, \psi_{(i-1)n_i})$ to be the only parent of (i, ψ_{ik}).

Since $\mathcal{T}$ is an infinite tree, by König's Lemma, $\mathcal{T}$ has an infinite branch $(0, \varphi)\ (1, \psi_{1l_1})\ (2, \psi_{2l_2})\ \ldots$. Because $a_i \wedge g_i \wedge rst(X_{il_i}) \wedge unch(Y_{il_i}) \wedge \bigcirc(\psi_{il_i})$ is a basic conjunction of $\beta(\psi_{(i-1)l_{(i-1)}})$ and $X_{il_i} \subseteq r_i \subseteq (X \backslash Y_{il_i})$, thus $(\psi_{(i-1)l_{(i-1)}}, a_i, g_i, r_i, \psi_{il_i})$ is a transition of $\mathbb{A}_\varphi$. Then from the fact that $v_i \models g_i$ and $v_{i+1} = (v_i[r_i]) + (t_{i+1} - t_i)$, we know that $(\varphi, \overline{0}) \xrightarrow{t_1} (\varphi, v_1) \xrightarrow{a_1} (\psi_{1l_1}, v_1[r_1]) \xrightarrow{t_2 - t_1} (\psi_{1l_1}, v_2) \xrightarrow{a_2} (\psi_{2l_2}, v_2[r_2]) \xrightarrow{t_3 - t_2} (\psi_{2l_2}, v_3) \xrightarrow{a_3} (\psi_{3l_3}, v_3[r_3]) \ldots$ is a run of $\mathbb{A}_\varphi$ that accepts $w = (t_1, a_1), (t_2, a_2), (t_3, a_3), \ldots$. □

Theorem 6 *Let φ be a co-safety MTL$_{0,\infty}$ formula over Σ, then $\mathcal{L}_{reach}(\mathcal{A}^u_\varphi, \{true\}) \backslash Zeno(\Sigma) \subseteq \mathcal{L}_{reach}(\mathbb{A}_\varphi, \{true\}) \backslash Zeno(\Sigma)$.*

Theorem 6 can be proved similarly as Theorem 5.

Example. The automaton in Fig. 3 is a deterministic under-approximation timed automaton for $\psi = (\Diamond_{\geq 1} a) \vee \Box_{\leq 2} b$.

5 Finding a winning strategy using UPPAAL-TIGA

Let us first consider that the specification contains exactly one requirement and one assumption, i.e. we are solving the synthesis problem $(\varphi \rightarrow \psi, \Sigma_c, \Sigma_u)$, where φ and ψ are safety-MTL$_{0,\infty}$ formulas over $\Sigma_c \cup \Sigma_u$.

The negation $\neg\varphi$ is in co-safety-MTL$_{0,\infty}$ and we apply the algorithm described in the previous section to build the deterministic TA $A_{\neg\varphi}$ such that $\mathcal{L}_{reach}(A_{\neg\varphi}, \{true\}) \backslash Zeno(\Sigma_c \cup \Sigma_u) \subseteq \mathcal{L}(\neg\varphi)$. ψ is in safety-MTL$_{0,\infty}$ and we build deterministic TA A_ψ such that $\mathcal{L}(A_\psi) \backslash Zeno(\Sigma_c \cup \Sigma_u) \subseteq \mathcal{L}(\psi)$. Next we add a location $false$ to both $A_{\neg\varphi}$ and A_ψ and add the transitions to it that are enabled when none of other outgoing transitions from the current location is enabled. In other words, we make A_ψ and $A_{\neg\varphi}$ complete so that they end up in $false$ if they cannot accept the input word. This syntactic transformation can be

done in linear time in the automaton size. In the rest of this section we describe how we apply UPPAAL- TIGA to $A_{\neg\varphi}$ and A_{ψ} to generate a solution for the synthesis problem.

UPPAAL- TIGA [5] is a tool that checks whether there exists a strategy for the controller to satisfy the given reachability or safety winning condition for all uncontrollable moves of the opponent player. Such a strategy can be syntactically represented by a timed automaton S derived from the input timed automata A where the guards of the controllable transitions are restricted. The semantics of strategy TA S is urgent for the controller, i.e. a controller always proposes to take the earliest enabled controllable transition. Thus this semantics is essentially the same as we gave in definition 4 when we described how a timed automaton implements a strategy for a synthesis problem.

Thus if A is deterministic and complete, and together with the winning condition it encodes the input specification $\varphi \rightarrow \psi$, then the winning strategy generated by UPPAAL- TIGA implements a solution for the synthesis problem $(\varphi \rightarrow \psi, \Sigma_c, \Sigma_u)$.

We construct such A by forming a parallel composition of $A_{\neg\varphi}$ and A_{ψ}, i.e. $A \equiv A_{\neg\varphi}||A_{\psi}$. This parallel composition is synchronized on actions and time delays, i.e. a transition $(s_1, s_2) \xrightarrow{d} (s_1', s_2')$ exists in A for some $d \in \mathbb{R}_{\geq 0} \cup \Sigma_c \cup \Sigma_u$ iff there exists a transition $s_1 \xrightarrow{d} s_1'$ in $A_{\neg\varphi}$ and there exists a transition $s_2 \xrightarrow{d} s_2'$ in A_{ψ}. Let $L^{\neg\varphi}_{true}$ be the set of locations of this parallel composition such that $A_{\neg\varphi}$ is in its location $true$, and let L^{ψ}_{false} be the set of locations such that A_{ψ} is in its location $false$.

The goal of the controller in $A_{\neg\varphi}||A_{\psi}$ in UPPAAL- TIGA is to:

- violate the assumption φ by visiting a location from $L^{\neg\varphi}_{true}$, or
- satisfy the requirement ψ by avoiding visiting the locations from L^{ψ}_{false}.

Unfortunately UPPAAL- TIGA does not support winning conditions of the form $\Diamond(L^{\neg\varphi}_{true}) \vee \Box(\neg L^{\psi}_{false})$. Thus we use a stricter winning condition of $(L^{\varphi}_{true})\mathrm{R}(\neg L^{\psi}_{false})$, that is more hard to control for the controller. Such variations of the winning condition was previously proposed in [8] where the authors study "strict realizability of the implication" $(\Diamond(L^{\neg\varphi}_{true}) \vee \Box(\neg L^{\psi}_{false}))$ and "realizability of the implication" $((L^{\varphi}_{true})\mathrm{R}(\neg L^{\psi}_{false}))$ for GR(1).

If the input specification contains more than one assumption and/or requirement, i.e it is of the form $\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \rightarrow \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_m$, then we can still consider $\varphi = \varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n$ as a single assumption and $\psi = \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_m$ as a single requirement (since safety-MTL$_{0,\infty}$ is closed under conjunction). However, in our implementation we construct a separate automaton for every formula, and search for a winning strategy in the parallel composition $A_{\neg\varphi_1}|| \cdots ||A_{\neg\varphi_n}||A_{\psi_1}|| \cdots ||A_{\psi_m}$. It is more efficient since we can avoid the full exploration of the game state space thanks to the on-the-fly game solving algorithm used by UPPAAL- TIGA.

Additionally, we have to make sure that the generated strategy of UPPAAL- TIGA is non-Zeno (otherwise, we cannot guarantee that monitoring TA approximates the specification due to Theorems 1 and 2). We do this in either of two different ways.

The first way is to try to prove that all Zeno strategies are losing for the controller. We do this by adapting the construction of [22]. And second, we can force UPPAAL- TIGA to generate a non-Zeno strategy by using a Büchi acceptance condition [15]. The second way requires a more expensive algorithm for the solution of timed games with Büchi objectives, thus it is advisable to try the first way first. We describe both approaches in details in the next subsection.

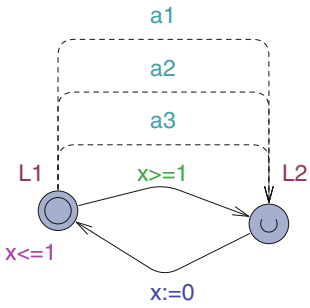

Fig. 4 A test automaton A_Z for the detection and forcing of non-Zeno behavior

5.1 Avoiding and detecting Zeno loops in UPPAAL- TIGA

Our goal is to avoid generating strategies that can produce the timed words from $Zeno(\Sigma_c)\backslash Zeno(\Sigma_u)$, we call such strategies non-Zeno.

Let us first describe the way of proving that all Zeno strategies are losing for the controller. We say that there is a controllable Zeno loop in $A_{\neg\varphi}||A_\psi$ iff there exists a run $s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{a_1} s_2 \dots$ such that this run is Zeno, all its actions are controllable starting from some point, and the run does not visit locations from L^ψ_{false}. It is easy to see that if there are no controllable Zeno loops in $A_{\neg\varphi}||A_\psi$, then there are also no winning Zeno strategies in it. In order to detect controllable Zeno loops, we put $A_{\neg\varphi}||A_\psi$ in parallel with a test automaton depicted in Fig. 4 (we consider here that $\Sigma_u = \{\mathbf{a1}, \mathbf{a2}, \mathbf{a3}\}$). This test automaton A_Z has two locations **L1** and **L2**, and location **L2** is urgent (see [6]), i.e. automaton arrives in location **L2**, it should leave it at the next transition without time being elapsed. Additionally, the location **L1** has an invariant $x \leq 1$ which states that the TA should leave this location as soon as the value of x reaches 1. We use UPPAAL (basic version) to check that $A_{\neg\varphi}||A_\psi||A_Z$ satisfies property $\Box(L1 \rightarrow \Diamond L2)$, that can be expressed as `L1 -> L2` in UPPAAL. It is easy to see that there exists a controllable Zeno loop in $A_{\neg\varphi}||A_\psi$ iff this property is not satisfied by $A_{\neg\varphi}||A_\psi||A_Z$.

If there exists a controllable Zeno loop in $A_{\neg\varphi}||A_\psi$, then we prevent the controller from playing a Zeno strategy by adding a requirement that the time should always progress. We do this by again considering a parallel composition $A_{\neg\varphi}||A_\psi||A_Z$ and asking UPPAAL- TIGA to find a winning strategy for the winning condition $((L^\varphi_{true})\mathrm{R}(\neg L^\psi_{false})) \wedge \Box\Diamond L2$ (see [15]).

6 Case studies

We present three case studies, and for some of them we compare the performance of our tool to the performance of Acacia+ [9] and Unbeast [18] that are state-of-the art synthesis tools for LTL. The comparison with Unbeast is of special interest because it is designed for the assume-guarantee properties, just like our approach. Acacia+ is especially efficient for the compositional specifications being of the form $\bigwedge_i \phi$, thus we use the compositional synthesis in Acacia+ and transform the specifications $\varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \psi_1 \wedge \dots \wedge \psi_m$ into $(\varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \psi_1) \wedge \dots \wedge (\varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \psi_m)$. We experimentally checked that Acacia+ is more efficient in this case, even if the input specification is larger. For Acacia+ and Unbeast we use the LTL3BA tool [4] as it gives smaller automata for our specifications than the LTL2BA tool.

For all the case studies our implementation managed to produce exact monitoring Timed Automata for the input specifications. Additionally, our implementation detected for all the

Table 1 Specification for the job scheduling problem

Assumption φ_1	The jobs do not to arrive too often $\square(job \rightarrow (\widehat{\square}_{\leq 1} \neg job))$
Requirement ψ_1	A computational unit should be assigned immediately to every incoming job $\square(job \rightarrow ((\neg job)\widehat{\mathrm{U}}_{\leq 0}(\bigvee_{i=1\ldots N} u_i)))$
Requirement ψ_2	The assignments should be preceded by requests $\square((\bigvee_{i=1\ldots N} u_i) \rightarrow \neg \mathrm{X}((\neg job)\mathrm{U}(\bigvee_{i=1\ldots N} u_i))) \wedge \neg(\bigvee_{i=1\ldots N} u_i)$
Requirement ψ_3	There should be a time gap of N between the assignments of the same computation unit $\bigwedge_{i=1\ldots N} \square(u_i \rightarrow (\widehat{\square}_{\leq N} \neg u_i))$

Table 2 Results for the job scheduling problem

N		2	3	4	5	6	7	8
Time	Zeno behavior is proved to be losing	<1s	<1s	<1s	<1s	7s	4m44s	*Timeout*
	Zeno behavior is avoided by Büchi	<1s	<1s	4s	1m38s	*Timeout*	*Timeout*	*Timeout*
Strategy size		12	28	60	124	252	508	–

specifications that there are no winning Zeno strategies (the timing results include this check). However, for the comparison reasons we also present the results for the Büchi-based approach. We report the number of rules (transitions) of a strategy graph as a strategy size.

The experiments were held in Amazon Elastic Cloud on a High-Memory Quadruple Extra Large Instance (64 GB of memory, processor power equivalent to 3.25 GHz 2007 Xeon processor). The timeout is fixed to 1 h.

6.1 Job scheduling problem

Consider the following online job scheduling problem. There are N computational units, and computation of a job on a single unit takes T time units. New jobs can arrive sporadically, and we assume that the minimal time between the arrivals of two jobs is 1 time unit.

The goal of the scheduler is to assign new jobs to the computational units, so no two jobs overlap in time on the same computational unit. We require that this assignment should happen immediately after a job arrives, and jobs cannot be moved from one computational unit to another.

Obviously, the problem has a solution iff $N \geq T$. We study the borderline case of $N = T$. Our specification is defined over a set of uncontrollable actions $\Sigma_u = \{job\}$ and a set of controllable actions $\Sigma_c = \{u_1, u_2, \ldots, u_N\}$. The environment triggers action job, when a new job arrives, and the controller can assign a computation unit i to it by triggering u_i. The specification is $\varphi_1 \rightarrow \psi_1 \wedge \psi_2 \wedge \psi_3$, where the components are defined in Table 1. The time results are given in Table 2. This specification is essentially based on time, thus we didn't apply Acacia+ and Unbeast to it. The strategy sizes are equal for the two ways of avoiding Zeno behavior in our implementation (although we don't prove that it is true in general).

Figure 5 contains solution for the $N = 2$. For simplicity, we removed one extra clock whose value is always equal to the value of x.

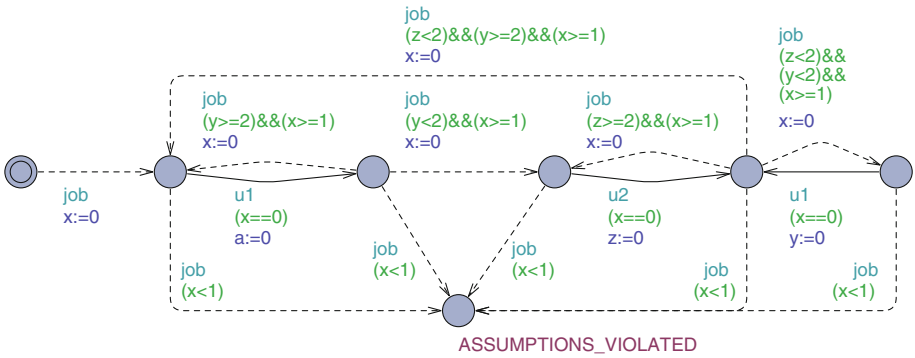

Fig. 5 Synthesized strategy for the jobs scheduling problem for $N = 2$

6.2 Dining Philosophers problem

Consider that there are N philosophers (that are a part of the environment). A philosopher may indicate that he is hungry (event $hungry_i$ for philosopher i). The controller can tell a philosopher to take his left or right fork, or tell him to start eating (events $lfork_i$, $rfork_i$ and eat_i for philosopher i). A philosopher may indicate that he finished eating (event $finish_i$), and at the same time he puts both forks on the table. Thus the set of uncontrollable actions is $\Sigma_u = \{hungry_1, \ldots, hungry_N, finish_1, \ldots, finish_N\}$, and the set of controllable actions is $\Sigma_c = \{lfork_1, \ldots, lfork_N, rfork_1, \ldots, rfork_N, eat_1, \ldots, eat_N\}$.

We also transformed this $\text{MTL}_{0,\infty}$ specification into the untimed LTL specifications for Acacia+ and Unbeast (basically, we removed all the timing constraints from the formula). As it can been seen from Table 6, Acacia+ and Unbeast scaled only up to $N = 2$, while our tool scaled up to $N = 4$. One can argue that the LTL specification for certainly requires Büchi acceptance condition, while our $\text{MTL}_{0,\infty}$ specification is a Boolean combination of safety formulas. Therefore we made another LTL specification for Acacia+ and Unbeast where we discretized time (i.e. assumed bounded variability), and this specification does not contain until operators and thus does not require Büchi acceptance condition. For this specification Acacia+ can also handle only the case of $N = 2$ with the maximal time granularity of 2, and Unbeast can additionally handle $N = 2$ and time granularity 3. We do not report strategy sizes for Unbeast because we were not able to extract them.

Our specification is given in Table 3, and the experimental results are presented in Table 6. The Acacia+ and Unbeast LTL specifications for the Dining Philosophers problem are described in details in the next two subsections.

6.2.1 Dining Philosophers specification for Acacia+ and Unbeast: LTL specification

The synthesis game in Acacia+ and Unbeast is played in rounds, and in each round one player (i.e. controller or environment) always plays first, and the other (second) player plays second. In each round the first player proposes some subset Σ_1 of his propositions, and the second player observes Σ_1 and proposes some subset Σ_2 of his propositions. Thus together they produce the next element of a run that is $\Sigma_1 \cup \Sigma_2$.

Thus in total we face three imprecisions when we translate our Dining philosophers specification from $\text{MTL}_{0,\infty}$ into LTL:

Table 3 Specification for the dining philosophers problem

Assumption $\varphi_\mathbf{1}$	A philosopher cannot say too often that he is hungry $\bigwedge_{i=1\ldots N} \square(hungry_i \rightarrow (\widehat{\square}_{<N} \neg hungry_i))$
Assumption $\varphi_\mathbf{2}$	A philosopher cannot say that he is hungry while he is eating, and he should finish eating within 1 time units $\bigwedge_{i=1\ldots N} \square(eat_i \rightarrow ((\neg hungry_i)\, \mathrm{U}_{<1}\, finish_i)))$
Assumption $\varphi_\mathbf{3}$	A philosopher cannot finish eating without starting eating $\bigwedge_{i=1\ldots N} \square(finish_i \rightarrow \neg\bigcirc((\neg eat_i)\, \mathrm{U}\, finish_i))) \wedge \bigwedge_{i=1\ldots N} \neg((\neg eat_i)\, \mathrm{U}\, finish_i)$
Requirement $\psi_\mathbf{1}$	A philosopher will start eating within $N-1$ time units after he became hungry $\bigwedge_{i=1\ldots N} \square(hungry_i \rightarrow ((\neg hungry_i)\widehat{\mathrm{U}}_{\leq N-1} eat_i)))$
Requirement $\psi_\mathbf{2}$	Once a fork is taken, it should be not taken again until it is put on the table $\bigwedge_{i=1\ldots N} \square((rfork_i \vee lfork_{i+1\,\%N}) \rightarrow$ $\neg\bigcirc((\neg finish_i \wedge \neg finish_{i+1\,\%N})\, \mathrm{U}\, (rfork_i \vee lfork_{i+1\,\%N})))$
Requirement $\psi_\mathbf{3}$	A philosopher cannot start eating if he does not hold his left fork $\bigwedge_{i=1\ldots N} \square(eat_i \rightarrow \neg\bigcirc((\neg lfork_i)\, \mathrm{U}\, eat_i))) \wedge \bigwedge_{i=1\ldots N} \neg((\neg lfork_i)\, \mathrm{U}\, eat_i)$
Requirement $\psi_\mathbf{4}$	A philosopher cannot start eating if he does not hold his right fork $\bigwedge_{i=1\ldots N} \square(eat_i \rightarrow \neg\bigcirc((\neg rfork_i)\, \mathrm{U}\, eat_i))) \wedge \bigwedge_{i=1\ldots N} \neg((\neg rfork_i)\, \mathrm{U}\, eat_i)$

- Our tool supports time constraints, while Acacia+ and Unbeast do not. Thus we drop all the timing constraints in the LTL specification. Additionally, in LTL we are not available to specify the fact that a philosopher cannot say $hungry$ too often (assumption $\varphi_\mathbf{1}$). Thus instead of saying this we use an assumption that a philosopher cannot say $hungry$ twice without being eating in between. We also consistently change $\varphi_\mathbf{2}$ and $\psi_\mathbf{1}$ to reflect this fact.
- At each round in LTL synthesis game each player can propose a set of propositions, not just a single action. Although we can force each player propose not more than one proposition at a time (by adding additional requirements and assumptions), it will make it more difficult for Acacia+ and Unbeast to synthesize a controller due to its algorithm that counts the number of times an automaton visits the Büchi acceptance locations. Thus we let each player to propose several propositions at once (e.g. tell two different philosophers to start eating). However, we do not allow a philosopher to say that he is hungry and say that he finished eating at the same time slot, since the order of these events is crucial and it is unspecified in this case (assumption $\varphi_\mathbf{4}$). Similarly, a controller cannot tell a philosopher to start eating and to take a fork at the same time (requirement $\psi_\mathbf{5}$). Additionally, we have to modify the requirement $\psi_\mathbf{2}$ to prevent the controller from giving the same fork simultaneously to two neighboring philosophers.
- The notion of the next state in our turn-based semantics is different for the first and the second players. We assume that at each round the environment is making the first move (by using the option "`-p 1`" of Acacia+). We also modify $\varphi_\mathbf{3}$ to handle the fact that in the semantics of Acacia+ and Unbeast the propositions of the controller and the environment are mixed together at each round.

LTL specification for the Dining Philosophers problem is given in Table 4.

6.2.2 Dining Philosophers specification in LTL using time discretization

We define a discretized version of Until operator:

Table 4 Untimed LTL specification for the dining philosophers problem

Assumptions φ_1	A philosopher cannot say that he is hungry twice without finishing eating $\bigwedge_{i=1\ldots N} \square(hungry_i \rightarrow \neg\bigcirc((\neg finish_i)\,\mathrm{U}\, hungry_i)))$
Assumptions φ_2	A philosopher cannot say that he is hungry while he is eating, and he should eventually finish eating $\bigwedge_{i=1\ldots N} \square(eat_i \rightarrow \bigcirc\Diamond finish_i))$
Assumptions φ_3	A philosopher cannot finish eating without starting eating $\bigwedge_{i=1\ldots N} \square(finish_i \rightarrow (eat_i \vee \neg\bigcirc((\neg eat_i)\,\mathrm{U}\, finish_i))) \wedge \bigwedge_{i=1\ldots N} \neg((\neg eat_i)\,\mathrm{U}\, finish_i)$
Assumptions φ_4	$finish_i$ and $hungry_i$ cannot be emitted simultaneously for the same i $\square(\neg(hungry_i \wedge finish_i))$
Requirements ψ_1	A philosopher will eventually eat after he is hungry $\bigwedge_{i=1\ldots N} \square(hungry_i \rightarrow \Diamond eat_i)$
Requirements ψ_2	Once a fork is taken by a philosopher, it should be not taken again until it is put on the table (here $putfork_i$ is an abbreviation for $finish_i \vee finish_{i+1\,\%N}$, for simplicity) $\bigwedge_{i=1\ldots N} \square((rfork_i) \rightarrow (\neg lfork_{i+1\,\%N}) \wedge \neg\bigcirc((\neg putfork_i)\mathrm{U}\,((\neg putfork_i) \wedge (rfork_i \vee lfork_{i+1\,\%N})))) \wedge$ $\bigwedge_{i=1\ldots N} \square((lfork_{i+1\,\%N}) \rightarrow (\neg rfork_i) \wedge \neg\bigcirc((\neg putfork_i)\mathrm{U}\,((\neg putfork_i) \wedge (rfork_i \vee lfork_{i+1\,\%N}))))$
Requirements ψ_3	A philosopher cannot start eating if he does not hold his left fork $\bigwedge_{i=1\ldots N} \square(eat_i \rightarrow \neg\bigcirc((\neg lfork_i)\,\mathrm{U}\, eat_i))) \wedge \bigwedge_{i=1\ldots N} \neg((\neg lfork_i)\,\mathrm{U}\, eat_i)$
Requirements ψ_4	A philosopher cannot start eating if he does not hold his right fork $\bigwedge_{i=1\ldots N} \square(eat_i \rightarrow \neg\bigcirc((\neg rfork_i)\,\mathrm{U}\, eat_i))) \wedge \bigwedge_{i=1\ldots N} \neg((\neg rfork_i)\,\mathrm{U}\, eat_i)$
Requirements ψ_5	We cannot emit eat_i and $lfork_i$ (or $rfork_i$) simultaneously for the same i $\square(\neg(eat_i \wedge (lfork_i \vee rfork_i))$

$$\varphi \mathrm{U}^*_{\leq d} \psi \equiv \psi \vee (\varphi \wedge \bigcirc\psi) \vee \cdots \vee (\varphi \wedge \bigcirc\varphi \wedge \cdots \wedge \underbrace{\bigcirc \ldots \bigcirc}_{d-1} \varphi \wedge \underbrace{\bigcirc \ldots \bigcirc}_{d} \psi).$$

We define $\Diamond^*_{\leq d}\varphi$ to be equal to $true\mathrm{U}^*_{\leq d}\phi$, and $\square^*_{\leq d}\varphi$ to be equal to $\neg\Diamond^*_{\leq d}\neg\varphi$.

Like in the previous (untimed) specification, in our LTL specification for fixed time granularity we also assume that at each slot(round) the environment plays first. Thus we use φ_4 and ψ_5 from the previous untimed specification.

Let us fix an integer k being a time granularity (i.e. a number of times a signal can change during one time unit). Our LTL specification for the Dining Philosophers problem with discretized time is given in Table 5.

6.3 Generalized buffer controller

The generalized buffer controller synthesis problem has been first studied in [7] for Generalized Reactivity (1). In [21] the authors reformulate this problem in LTL and apply Acacia+ to it. This specification describes a system that contains several senders and receivers with a buffer in-between them. The specification is of the form $(\varphi_s \rightarrow \psi_s) \wedge (\varphi_r \rightarrow \psi_r)$, where φ_s (φ_r, correspondingly) is the assumption over the behavior of the senders (receivers) and ψ_s (ψ_r, correspondingly) is the requirements over the behavior of the controller with respect to senders (receivers). We transformed the specification into $(\varphi_s \wedge \varphi_r) \rightarrow (\psi_s \wedge \psi_r)$, since the former specification $(\varphi_s \rightarrow \psi_s) \wedge (\varphi_r \rightarrow \psi_r)$ is not supported by our tool. Thus in the experiments for our tool we study a weaker specification than Acacia+'s, and our $\mathrm{MTL}_{0,\infty}$

Table 5 LTL specification for the dining philosophers problem with discretized time

Assumption φ_1	A philosopher cannot say too often that he is hungry $\bigwedge_{i=1...N} \square(hungry_i \rightarrow \bigcirc \square^*_{\leq k-2}(\neg hungry_i))$
Assumption φ_2	A philosopher cannot say that he is hungry while he is eating, and he should finish eating within 1 time units $\bigwedge_{i=1...N} \square(eat_i \rightarrow ((\neg hungry_i)\ \mathrm{U}^*_{\leq k-1} finish_i)))$
Assumption φ_3	A philosopher cannot finish eating without starting eating $\bigwedge_{i=1...N} \square(finish_i \rightarrow (eat_i \vee \neg\bigcirc((\neg eat_i)\ \mathrm{U}\ finish_i)))) \wedge \bigwedge_{i=1...N} \neg((\neg eat_i)\ \mathrm{U}\ finish_i)$
Assumptions φ_4	$finish_i$ and $hungry_i$ cannot be emitted simultaneously for the same i $\square(\neg(hungry_i \wedge finish_i))$
Requirement ψ_1	A philosopher will start eating within $N-1$ time units after he became hungry $\bigwedge_{i=1...N} \square(hungry_i \rightarrow ((\neg hungry_i)\mathrm{U}^*_{\leq k(N-1)} eat_i)))$
Requirements ψ_2	Once a fork is taken by a philosopher, it should be not taken again until it is put on the table (here $putfork_i$ is an abbreviation for $finish_i \vee finish_{i+1\,\%N}$, for simplicity) $\bigwedge_{i=1...N} \square((rfork_i) \rightarrow (\neg lfork_{i+1\,\%N}) \wedge \neg\bigcirc((\neg putfork_i)\ \mathrm{U}\ ((\neg putfork_i) \wedge (rfork_i \vee lfork_{i+1\,\%N})))) \wedge$ $\bigwedge_{i=1...N} \square((lfork_{i+1\,\%N}) \rightarrow (\neg rfork_i) \wedge \neg\bigcirc((\neg putfork_i)\ \mathrm{U}\ ((\neg putfork_i) \wedge (rfork_i \vee lfork_{i+1\,\%N}))))$
Requirement ψ_3	A philosopher cannot start eating if he does not hold his left fork $\bigwedge_{i=1...N} \square(eat_i \rightarrow \neg\bigcirc((\neg lfork_i)\ \mathrm{U}\ eat_i))) \wedge \bigwedge_{i=1...N} \neg((\neg lfork_i)\ \mathrm{U}\ eat_i)$
Requirement ψ_4	A philosopher cannot start eating if he does not hold his right fork $\bigwedge_{i=1...N} \square(eat_i \rightarrow \neg\bigcirc((\neg rfork_i)\ \mathrm{U}\ eat_i))) \wedge \bigwedge_{i=1...N} \neg((\neg rfork_i)\ \mathrm{U}\ eat_i)$
Requirements ψ_5	We cannot emit eat_i and $lfork_i$ (or $rfork_i$) simultaneously for the same i $\square(\neg(eat_i \wedge (lfork_i \vee rfork_i))$

Table 6 Results for the dining philosophers problem

N	Our tool						
	Time		Strategy size				
	Zeno is proved to be losing	Zeno is avoided					
2	3s	11s	1931				
3	14s	*Timeout*	65868				
4	14m4s	*Timeout*	2107776				
N	Acacia+ (untimed)		Acacia+ (time granularity 2)		Unbeast (untimed)	Unbeast (time granularity 2)	Unbeast (time granularity 3)
	Time	Strategy size	Time	Strategy size	Time	Time	Time
2	2m12s	439	41s	212	1m13s	21s	2m53s

specification requires the buffer to behave correctly only if both senders and receivers follow their assumptions.

Consider that there are N senders and M receivers. Acacia+ specification is defined over controllable signals $\{b2s\_ack\_i\}$ and $\{b2r\_req\_j\}$ and uncontrollable signals $\{s2b\_req\_i\}$

Table 7 $\mathrm{MTL}_{0,\infty}$ specification for the generalized buffer controller problem of [21]

Assumption φ_1	*s2b_req_i_off* comes only after *s2b_req_i_on* $\bigwedge_{i=1\ldots N}(\Box(s2b\_req\_i\_off \rightarrow \neg\bigcirc((\neg s2b\_req\_i\_on)\mathrm{U}(s2b\_req\_i\_off)))) \wedge \neg((\neg s2b\_req\_i\_on)\mathrm{U}(s2b\_req\_i\_off))$
Assumption φ_2	A request is not lowered until it is served $\bigwedge_{i=1\ldots N}\Box(s2b\_req\_i\_on \rightarrow \neg((\neg b2s\_ack\_i)\mathrm{U}(s2b\_req\_i\_off)))$
Assumption φ_3	The sender should immediately deassert *s2b_req_i* after getting the acknowledgment $\bigwedge_{i=1\ldots N}\Box(b2s\_ack\_i \rightarrow \Diamond_{x\leq 0} s2b\_req\_i\_off)$
Assumption φ_4	Acknowledgment from the receiver can come only after the request from the buffer $\bigwedge_{j=1\ldots M}(\Box(r2b\_ack\_j \rightarrow \neg\bigcirc((\neg b2r\_req\_j\_on)\mathrm{U}(r2b\_ack\_j)))) \wedge \neg((\neg b2r\_req\_j\_on)\mathrm{U}(r2b\_ack\_j))$
Assumption φ_5	Acknowledgment from the receiver will come within 1 time units after the request from the buffer $\bigwedge_{j=1\ldots M}\Box(b2r\_req\_j\_on \rightarrow \Diamond_{x\leq 1} r2b\_ack\_j))$
Requirement ψ_1	*b2r_req_j_off* comes only after *b2r_req_j_on* $\bigwedge_{j=1\ldots M}(\Box(b2r\_req\_j\_off \rightarrow \neg\bigcirc((\neg b2r\_req\_i\_on)\mathrm{U}(b2r\_req\_i\_off)))) \wedge \neg((\neg b2r\_req\_i\_on)\mathrm{U}(b2r\_req\_i\_off))$
Requirement ψ_2	A buffer should immediately acknowledge a request from the sender $\bigwedge_{i=1\ldots N}\Box(s2b\_req\_i\_on \rightarrow \Diamond_{x\leq 0} b2s\_ack\_i))$
Requirement ψ_3	Acknowledgment from buffer can come only request from the sender $\bigwedge_{i=1\ldots N}(\Box(b2s\_ack\_i \rightarrow \neg\bigcirc((\neg s2b\_req\_i\_on)\mathrm{U}(b2s\_ack\_i)))) \wedge \neg((\neg s2b\_req\_i\_on)\mathrm{U}(b2s\_ack\_i))$
Requirement ψ_4	Only one sender sends data at any one time $\bigwedge_{i=1\ldots N}\Box(b2s\_ack\_i \rightarrow \neg((\neg s2b\_req\_i\_off)\mathrm{U}(\bigvee_{k\neq i} b2s\_ack\_k)))$
Requirement ψ_5	A request is not lowered until it is served $\bigwedge_{j=1\ldots M}\Box(b2r\_req\_j\_on \rightarrow \neg((\neg r2b\_ack\_j)\mathrm{U}(b2r\_req\_j\_off)))$
Requirement ψ_6	The buffer should immediately deassert *b2r_req_j* after getting the acknowledgment $\bigwedge_{j=1\ldots M}\Box(r2b\_ack\_j \rightarrow \Diamond_{x\leq 0} b2r\_req\_j\_off))$
Requirement ψ_7	The buffer will not make two consecutive requests to any receiver $\bigwedge_{j=1\ldots M}\Box(b2r\_req\_j\_on \rightarrow \neg\bigcirc((\neg\bigvee_{k\neq j} b2r\_req\_k\_on)\mathrm{U}(b2r\_req\_j\_on)))$
Requirement ψ_8	The buffer does not request two receivers simultaneously $\bigwedge_{j=1\ldots M}(b2r\_req\_j\_on \rightarrow \neg((\neg b2r\_req\_j\_off)\mathrm{U}(\bigvee_{k\neq j}(b2r\_req\_k\_on))))$

and $\{r2b\_ack\_j\}$ where i ranges in $1\ldots N$ and j ranges in $1\ldots M$. $\{b2r\_req\_j\}$ and $\{s2b\_req\_i\}$ are dealt as continuous signals that can be turned on for some duration of time. Thus in the specification for our tool we model each of these signals with two actions, one for the start of a signal and one for its end, i.e. we defined the actions *b2r_req_j_on*, *b2r_req_j_off* and *s2b_req_i_on*, *s2b_req_i_off*. The signal *b2r_req_j* (*s2b_req_i*) is assumed to be turned on in between the actions *b2r_req_j_on* and *b2r_req_j_off* (*s2b_req_i_on* and *s2b_req_i_off*). $\{b2s\_ack\_i\}$ and $\{r2b\_ack\_j\}$ are instantaneous signals, thus we leave them as actions as it is.

The goal of this case study is to demonstrate that our tool can be applied to a specification already used with another tool. Thus we do not translate the formula in the format of Unbeast and also use only the specifications sizes provided in the Acacia+ distribution.

Our specification is given in Table 7. Basically, we added time constraints to the unbounded "eventually" operators, so that the specification fits the format supported by our tool.

Table 8 contains the experimental results for this problem.

Table 8 Results for the generalized buffer specification

	Our tool			Acacia+	
	Time		Strategy size	Time	Strategy size
	Zeno is proved to be losing	Zeno is avoided by Büchi			
2 Senders, 2 receivers	2s	5s	18,871	<1s	219
2 Senders, 3 receivers	3s	13s	35,748	1s	317
2 Senders, 4 receivers	5s	26s	57,239	3s	417
2 Senders, 5 receivers	9s	48s	83,618	8s	744
2 Senders, 6 receivers	14s	1m23s	114,885	23s	955
2 Senders, 7 receivers	22s	2m13s	151,040	1m34s	1,067

7 Conclusions

We present an algorithm for the synthesis of the specifications of the form $\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n \rightarrow \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_m$, where all φ_i and ψ_j are safety-MTL$_{0,\infty}$ formulas. The algorithm is based on a novel translation procedure from safety and co-safety MTL$_{0,\infty}$ into under-approximation deterministic timed automata. This allows our approach to scale well to large specifications, and have a comparable or better performance to the state-of-the-art synthesis tools Acacia+ and Unbeast (the specifications are not precisely the same, but have been modified for a timed setting). We believe that our approach can be extended to the specifications being arbitrary Boolean combinations of safety-MTL$_{0,\infty}$ properties.

Our approach is sound but not complete. This incompleteness comes from the fact that the generated monitoring TA can be non-exact (under-approximates the input specification). Additionally we use the winning condition $(L^{\varphi}_{true})\mathrm{R}(\neg L^{\psi}_{false})$ in UPPAAL-TIGA that is stronger than the minimally required $\Diamond(L^{\neg\varphi}_{true}) \vee \Box(\neg L^{\psi}_{false})$. However, our tool managed to produce the solutions for all the specifications we studied. In the future work we may address the decidability of the original problem (i.e. is it possible to provide a complete algorithm), and provide an algorithm with better coverage.

References

1. Alur, R: Formal verification of hybrid systems. In: Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT '11), pap. 273–278. ACM, New York, NY, USA (2011)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**, 183–235 (1994)
3. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM **43**(1), 116–146 (1996)
4. Babiak, T., Kretínský, M., Rehák, V., Strejcek, J.: LTL to Büchi Automata Translation: Fast and More Deterministic. CoRR, abs/1201.0682 (2012)
5. Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K.G., Lime, D.: Uppaal-tiga: time for playing games! In: Proceedings of the 19th International Conference on Computer Aided Verification, Number 4590 in LNCS, pp. 121–125. Springer, Berlin (2007)
6. Behrmann, G., David, Re, Larsen, K.G.: A Tutorial on Uppaal. Springer, Berlin (2004)
7. Bloem, R., Galler, S., Jobstmann, B., Piterman, N., Pnueli, A., Weiglhofer, M.: Specify, compile, run: hardware from PSL. In: 6th International Workshop on Compiler Optimization Meets Compiler Verification (2007)
8. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012)

9. Bohy, A., Bruyère, V., Filiot, E., Jin, N., Raskin, J.-F.: Acacia+, a tool for LTL synthesis. In: Proceedings of the 24th International Conference on Computer Aided Verification, CAV'12, pp. 652–657. Springer, Berlin, Heidelberg (2012)
10. Bouyer, P., Bozzelli, L., Chevalier, F.: Controller synthesis for MTL specifications. In: Proceedings of the 17th International Conference on Concurrency Theory (CONCUR'06) (2006)
11. Buchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Trans. Am. Math. Soc. **138**, 295–311 (1969)
12. Bulychev, P., David, A., Larsen, K. G., Legay,A., Li, G., Poulsen, D. B., Stainer, A.: Monitor-based statistical model checking for weighted metrictemporal logic. In: LPAR (2012)
13. Cassez, F., David, A., Fleury, E., Larsen, K. G., Lime, D.: Efficient on-the-fly algorithms for the analysis of timed games. In: CONCUR'05, volume 3653 of LNCS, pp. 66–80.Springer, Berlin (August 2005)
14. Church, A.: Logic, Arithmetic. Automata. In: Proceedings of the International Mathematical Congress (1962)
15. David, A., Behrmann, G., Bulychev, P., Byg, J.,Chatain, T., Larsen, T.G., Pettersson, P., Rasmussen, J., Srba, J., Yi, W., Joergensen, K.Y.,Lime, D., Magnin, M., Roux, O.H., Traonouez, L.-M.: Tools for model-checking timed systems. In: RouxO.H., Claude, J. (eds.) Communicating Embedded Systems—Software and Design, pp. 165–225. ISTE Publishing, Wiley, New York (2009)
16. Di Giampaolo, B., Geeraerts, G, Raskin, J.F., Sznajder, N.: Safraless procedures for timed specifications. In: Springer (ed.) Proceedings of FORMATS 2010, 8th International Conference on Formal Modelling and Analysis of Timed Systems, volume 6246 of, Lecture Notes in Computer Science, pp. 2–22, (2010)
17. Doyen, L., Geeraerts, G., Raskin, J.F., Reicher, J.: Realizability of real-time logics. In: Proceedings of FORMATS 2009, 7th International Conference on Formal Modeling and Analysis of Timed Systems, volume 5813 of Lecture Notes in Computer Science, pp. 133–148. Springer, Berlin (2009)
18. Ehlers, R.: Symbolic bounded synthesis. In: Touili, T., Cook, B., Jackson, P. (ed.) 22nd International Conference on Computer Aided Verification, volume 6174 of LNCS, pp. 365–379. Springer, Berlin (2010)
19. Emerson, E.A., Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons. Sci. Comput. Program. **2**(3), 241–266 (1982)
20. Filiot, E., Jin, N., Raskin, J.-F.: An antichain algorithm for LTL realizability. In: CAV, pp. 263–277 (2009)
21. Filiot, E., Jin, N., Raskin, J.-F.: Exploiting structure in LTL synthesis. Int. J. Softw. Tools Technol. Transf. (STTT) 541–561 (2013). doi:10.1007/s10009-012-0222-5
22. Gómez, R., Bowman, H.: Efficient detection of zeno runsin timed automata. In: Proceedings of the 5th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS'07, pp. 195–210. Springer, Berlin, Heidelberg (2007)
23. Kupferman, O., Piterman, N., Vardi, M.Y.: Safraless compositional synthesis. In: 18th Conference on Computer Aided Verification, pp. 31–44 (2006)
24. Kupferman, O., Vardi, M.Y.: μ-Calculus synthesis. In: MFCS, pp. 497–507 (2000)
25. Maler, O., Nickovic, D., Pnueli, A.: Real time temporal logic: past, present, future. In: FORMATS, pp. 2–16 (2005)
26. Maler, O., Nickovic, D., Pnueli, A.: On synthesizing controllers from bounded-response properties. In: CAV, pp. 95–107 (2007)
27. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems. In: Mayr, E.W., Puech, C. (eds.) Proceedings of the STACS'95, LNCS 900, pp. 229–242. Springer, Berlin (1995)
28. Manna, Z., Wolper, P.: Synthesis of communicating processes from temporal logic specifications. ACM Trans. Program. Lang. Syst. **6**(1), 68–93 (1984)
29. Ouaknine, J., Worrell, J.: On the decidability of metric temporal logic. In: Proceedings of the 20th Annual IEEE Symposium onLogic in Computer Science, LICS '05, pp. 188–197. IEEE Computer Society, Washington, DC, USA (2005)
30. Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. In: Proceedings of the Verification, Model Checking, and Abstract Interpretation (VMCAI 06), pp. 364–380. Springer, Berlin (2006)
31. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In:Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on principlesof programming languages (POPL '89), pp. 179–190. ACM, New York, NY, USA (1989)
32. Ramadge, P., Wonham, W.: Supervisory control of a class of discrete event processes. SIAM J. Control Optim. **25**(1), 206–230 (1987)
33. Schewe, S., Finkbeiner, B.: Bounded synthesis. In: ATVA, volume 4762 of Lecture Notes in Computer Science, pp. 474–488. Springer, Berlin (2007)