Practical Controller Synthesis for $\text{MTL}_{0,\infty}$

Guangyuan Li
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China

Peter Gjøl Jensen
Computer Science, Aalborg University, Denmark

Kim Guldstrand Larsen
Computer Science, Aalborg University, Denmark

Axel Legay
INRIA, Rennes, France

Danny Bøgsted Poulsen
Christian-Albrechts University, Kiel, Germany

ABSTRACT

Metric Temporal Logic $\text{MTL}_{0,\infty}$ is a timed extension of linear temporal logic, LTL, with time intervals whose left endpoints are zero or whose right endpoints are infinity. Whereas the satisfiability and model-checking problems for $\text{MTL}_{0,\infty}$ are both decidable, we note that the controller synthesis problem for $\text{MTL}_{0,\infty}$ is unfortunately undecidable. As a remedy of this we propose an approximate method to the synthesis problem, which we demonstrate to be adequate and scalable to practical examples. We define a method for converting $\text{MTL}_{0,\infty}$ formulas into (nondeterministic) Timed Game Büchi Automata and furthermore show how to construct determinized over- and underapproximation of a such. For the proposed method, we present a toolchain seamlessly integrating the needed components for practical $\text{MTL}_{0,\infty}$ synthesis. Lastly we demonstrate on a number of case-studies the applicability and scalability of the proposed method.

1 INTRODUCTION

Automatic controller synthesis offers the promise of a disruptive technology for developing correct-by-construction control software. In short, 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 [10] and then theoretically solved for various specification formalisms in [7] and later works [5, 13, 17, 21, 23, 24].

The synthesis problem is computationally harder for linear time logics than the satisfiability and model-checking problems, and was for this reason considered intractable for a long time. Until recently, the intractability of proposed methods stemmed from the determinization of Büchi automata, which is a computationally hard problem. However, the synthesis problem has recently gained in practical performance due to the development of the so-called Safraless synthesis algorithms [16] that avoid the Büchi determinization phase. For real-time specification, the Metric Interval Temporal Logic (MITL [3]) is a logic that has proven its usefulness for speciciations [1] and thus a logic adequate synthesing controllers for. Unfortunately, the synthesis problem is known to be undecidable [11] for general MITL - but restricting the formulas to certain sub-classes the synthesis problem is rendered decidable [5, 19]. Overall, the main challenge in the real-time setting is that the Safraless approach is not always applicable as determinization is not possible in general. Allowing only upper or lower bounds on all until operators gives a sub-class of MITL called $\text{MTL}_{0,\infty}$. Although the satisfiability and model-checking problems for $\text{MTL}_{0,\infty}$ are both decidable, the controller synthesis problem is still rendered undecidable – this follows trivially from the work on Event Clock Logic by Doyen et. al [11], as we will show later in this paper. However, it is still possible to synthesise controllers for some $\text{MTL}_{0,\infty}$ formulas by use of an approximate technique – such as that we here present.

The main obstacle for synthesising controller for a $\text{MTL}_{0,\infty}$ objective is the construction of a Deterministic Timed Büchi Automata equivalent to the objective. Unfortunately, in a previous work [8], we already argued that the sub-class with only upper bounds ($MTL_{\leq a}$) is non-deterministic in the sense that for some formulas no Deterministic Timed Büchi Automaton exists. In that work, we showed how to construct over- and under-approximating automata for a given specification. The construction was implemented in the tool Casaal and used for monitoring purposes. Furthermore, experimental results witnessed that the approximations were often exact and when not exact, at least tight. The often "exact and tight" propertys of our previous work gave hope that a similar construction could be made for the full class of $\text{MTL}_{0,\infty}$ formulas and used for controller synthesis. The idea is to parallel compose the automaton into the model of the environment and obtain a Timed Game with Büchi Objectives - a tool like Uppaal-Tiga [4] can then be used to synthesise the controller. For the cases where an deterministic and exact Büchi automaton does not exist for the objective, the under-approximation may be used instead to construct a safe controller. For the purpose of synthesis for the over-approximation is mainly to verify the non-existence of a controller i.e. if you cannot synthesise a controller for the over-approximation you cannot synthesise one for the original objective.

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

In the current paper we show how to construct under- and over-approximation for $\text{MTL}_{0,\infty}$ objectives and we extend CASAAL for this new construction. Experiments show that in many cases the approximations are in fact exact. Our main contribution is the approach for synthesising controllers for $\text{MTL}_{0,\infty}$ objectives, but along the way we also develop the – to our knowledge – first exact translation from $\text{MTL}_{0,\infty}$ to (non-deterministic) Timed Büchi Automata. That particular construction is since modified – using techniques developed in a previous work [8] – to obtain the final under-/over-approximating deterministic Timed Büchi Automata. Another contribution of the paper is a tool chain that seamlessly integrate CASAAL and UPPAAL-TIGA [4] to form a practical way of synthesising controllers for $\text{MTL}_{0,\infty}$ objectives. We also demonstrate the applicability of our method on a number of case-studies, showing that the synthesis of controllers for $\text{MTL}_{0,\infty}$ objectives is feasible withing a reasonable computation time for non-trivial formulas and reasonable model-sizes. Our experiments demonstrate that the over- and underapproximation is often exact, supporting our claim of a "exact and tight" property. In short, our contributions are

- full and exact translation $\text{MTL}_{0,\infty}$ objectives into (non-deterministic) Timed Büchi Automata,
- automatic construction of deterministic over- and underapproximations, implemented in CASAAL,
- seamless integration between UPPAAL-TIGA and CASAAL in a single tool-chain for synthesis, and
- a demonstration of the approach on a number of case-studies.

The paper is structured in the following way: in Section 2, we introduce timed games and $\text{MTL}_{0,\infty}$. Section 3, proposes the translation from $\text{MTL}_{0,\infty}$ to (non-deterministic) Timed Büchi Automata. Section 4 presents the tool chain and demonstrates the applicability and efficiency of the tool chain through a number of practical examples. Full proofs are provided in an Appendix.

Related Work. The continuous semantics and the pointwise semantics are two commonly adopted semantics for MITL. Rajeev Alur et al. in [3] proposed a procedure for translating MITL (under continuous semantics) into timed Buchi automata, this procedure has never been implemented in practice. Oded Maler et al.[18] proposed a procedure to translate MITL(under continuous semantics) into temporal testers (not timed Büchi automata), their procedure also has not been implemented. Marc Geilen[14] has implemented a procedure to translate bounded $\text{MTL}_{0,\infty}$ to timed automata, the semantics he used is also the continuous semantics. As for pointwise semantics, in previous papers [9, 8], we have provided a constructions and a tool component (CASAAL) for translating the safety fragment and co-safety of $\text{MTL}_{0,\infty}$ into timed automata. In a recent paper [6], Thomas Brihaye et al. proposed a technique to translate MITL into Timed Büchi Automata through alternating timed automata. Their approach is based on a new (interval) semantics, where clock valuations are not real values, but intervals with real endpoints.

2 TIMED GAMES AND $\text{MTL}_{0,\infty}$

Let us introduce the main formalism and definitions used throughout the text. A timed word ω over a finite set of actions Σ is an infinite sequence of time points and actions $\omega = (t_1, a_1)(t_2, a_2)(t_3, a_3)\ldots$, where for every i we have $a_i \in \Sigma$, $t_i \in \mathbb{R}_{\geq 0}$ and $t_{i+1} \geq t_i$. A timed word $\omega = (t_1, a_1)(t_2, a_2)(t_3, a_3)\ldots$ is called non-Zeno if the sequence $\{t_i\}_{i\in\mathbb{N}}$ is unbounded.

Let X be a finite set of real-valued variables called clocks. A clock bound over X has the form $x \sim n$ or $x - y \sim n$, where $x, y \in X$, $\sim \in \{<, \leq, \geq, >\}$ and $n \in \mathbb{Z}_{\geq 0}$. We denote the set of all possible clock bounds over X by $\mathcal{B}(X)$, and let $\Theta(X)$ be the set of all Boolean formulas over $\mathcal{B}(X)$ (including conjunctions and disjunctions). A valuation over X is 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 $\delta \in \mathbb{R}_{\geq 0}$ we let $v + \delta$ to be the valuation such that $(v + \delta)(x) = v(x) + \delta$ for every clock $x \in X$; and $v[Y]$ is the valuation where $v[Y](x) = 0$ if $x \in Y$ and $v[Y](x) = v(x)$ otherwise.

Definition 2.1. A Timed Büchi Automaton (TBA) over actions Σ is a tuple $\mathcal{A} = (L, \ell_0, X, F, E)$, where L is a finite set of locations, ℓ_0 is the initial location, X is a finite set of clocks, $F \subseteq L$ is a set of accepting locations, and $E \subseteq L \times \Sigma \times \Theta(X) \times 2^X \times L$ is a set of edges.

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

A run ρ of a TBA $\mathcal{A}$ is an infinite sequence of alternating delay and discrete transitions $\rho = (l_0, \overline{0}) \xrightarrow{\delta_1} (l_1, v_1) \xrightarrow{a_1} (l_2, v_2) \xrightarrow{\delta_2} (l_3, v_3) \xrightarrow{a_2} (l_4, v_4) \xrightarrow{\delta_3} \ldots$. We say$\rho$ is accepting if $l_i \in F$ for infinitely many i. For $i \in \mathbb{N}$ we denote by ρ_i the finite prefix of ρ upto (l_i, v_i). We denote by $\mathit{Exec}(\mathcal{A})$ ($\mathit{Exec}^f(\mathcal{A})$) the set of all (finite) runs of $\mathcal{A}$. A timed word $\omega = (\delta_1, a_1)(\delta_1 + \delta_2, a_2)(\delta_1 + \delta_2 + \delta_3, a_3)\ldots$ is accepted by a TBA $\mathcal{A}$ if there exists an accepting run ρ for which ω is the corresponding timed word. We use $\mathcal{L}(\mathcal{A})$ to denote the set of all non-Zeno timed words accepted by $\mathcal{A}$. An ordinary Timed Automaton (TA) with final locations may be represented as a Timed Büchi Automaton by making all final locations terminal (looping) and accepting.

Definition 2.2. A Timed Game with Büchi conditions (TGB) over disjoint sets of controllable and uncontrollable actions, Σ_c and Σ_u, is a Timed Büchi Automaton $\mathcal{G} = (L, l_0, X, F, E)$ over $\Sigma_c \cup \Sigma_u$. A Timed Game (TG) is a TGB where all locations are accepting.

A strategy for a TGB $\mathcal{G}$ is a mapping σ, which given a finite run ρ describes how the run may proceed according to a controller. Formally $\sigma : \mathit{Exec}^f(\mathcal{G}) \to \Sigma_c \cup \{\lambda\}$, where

λ indicates a delay action. A strategy σ is only allowed to suggest actions allowed by the TGB and thus, given a finite run ρ ending in a state (l, v), (1) if $\sigma(\rho) = a \in \Sigma_c$, then there must exist a transition $(l, v) \xrightarrow{a} (l', v')$ and (2) if $\sigma(\rho) = \lambda$, there must exist a positive delay $\delta \in \mathbb{R}_>$ such that $(l, v) \xrightarrow{\delta} (l', v')$.

Given a strategy σ, we say that an infinite run $\rho = (l_0, \overline{0}) \xrightarrow{\delta_1} (l_1, v_1) \xrightarrow{a_1} (l_2, v_2) \xrightarrow{\delta_2} (l_3, v_3) \xrightarrow{a_2} (l_4, v_4) \xrightarrow{\delta_3} \dots$ is consistent with σ if for any $i \in \mathbb{N}$ either $(l_i, v_i) \xrightarrow{\alpha} (l_{i+1}, v_{i+1})$ and $(\alpha \in \Sigma_u) \vee ((\alpha \in \Sigma_c) \wedge \alpha = \sigma(\rho_i))$, or $\alpha = \delta \in \mathbb{R}_>$ and $\sigma(\rho_i \xrightarrow{\delta'}) = \lambda$ whenever $\delta' < \delta$. We denote by $Outcome(\mathcal{G}, \sigma)$ all runs that are consistent with σ, and denote by $\mathcal{L}(\mathcal{G}, \sigma)$ the corresponding set of timed words.

Given a TGB $\mathcal{G}$, we say that a strategy σ is winning if whenever $\rho \in Outcome(\mathcal{G}, \sigma)$, then ρ is accepting. Given a TG $\mathcal{G}$ and a set of timed words $\mathcal{L}$, we say that a strategy σ is winning with respect to the objective $\mathcal{L}$ if $\mathcal{L}(\mathcal{G}, \sigma) \subseteq \mathcal{L}$. When $\mathcal{L}$ is expressed using a deterministic TBA, the following easily obtained result is crucial for the method we develop in the following sections:

THEOREM 2.3. *Let $\mathcal{G}$ be a TG and $\mathcal{A}$ a determinstic TBA. Then $\mathcal{G}$ has a winning strategy with respect to $\mathcal{L}(\mathcal{A})$ if and only if the TGB $\mathcal{G} \otimes \mathcal{A}$ has a winning strategy*[1].

The emptiness problem for TBA is known to be PSPACE-complete [2] and the existence of winning strategies for TGB is EXPTIME-complete [20]. Moreover, for the synthesis problem, memoryless strategies suffices. The tools UPPAAL and UPPAAL-TIGA provide efficient on-the-fly exploration of a finite *symbolic reachability graph*, where the nodes are *symbolic states*. A symbolic state S is a pair (l, Z), where l is a location and Z is a so-called *zone* being the set of valuations satisfying a given clock constraint $g \in \mathcal{B}(X)$. In particular, a winning strategy σ produced by UPPAAL-TIGA for a given TGB is represented using zones. More precisely, for each location l, the representation R_σ gives a finite set of pairs $R_\sigma(l) = \{(Z_1, a_1), \dots, (Z_n, a_n)\}$, where $a_i \in \Sigma_c \cup \{\lambda\}$ with $Z_i \cap Z_j = \emptyset$ if $i \neq j$. Given a state (l, v) the value of the the strategy σ is simply a if $v \in Z$ with $(Z, a) \in R_\sigma(l)$.

Example 2.4. Consider the game in Fig. 1, where a *Cat* chases a *Mouse* on a 5×5 grid. Initially the *Cat* and the *Mouse* are in positions $(1, 5)$ and $(5, 1)$, respectively. During the chase, they may both repeatedly move to any legal neighbouring position (note that position $(3, 3)$ is illegal as there is already a flower-pot). Formally, the chasing game is modelled as a TG, being the product of a TG component for the *Cat* (controller) and a TG component for the *Mouse* (environment). For both the *Cat* and the *Mouse*, there is a minimum time-seperation between two consequitive moves, being 5 and 6 respectively. A simplest objective of the game is for the *Cat* to catch the *Mouse*, i.e. to bring the Timed Game into a product-state $(P^c_{i,j}, P^m_{i,j})$ for some legal position (i, j). More advanced objectives could be to ensure that the *Cat* will repeatedly catch the *Mouse*, and to do so within a maximum time-bound, say 40. In addition the *Cat* might for some reason want to repeatedly return to its initial position with some (minimum or maximum) time-seperation.

[1] $\mathcal{G} \otimes \mathcal{A}$ is the TGB obtained as a synchronous product of $\mathcal{G}$ and $\mathcal{A}$, where accepting states are determined by the $\mathcal{A}$ component.

2.1 Metric Temporal Logic $\text{MTL}_{0,\infty}$

Applying Theorem 2.3 it suffices to express the objectives of a TG as deterministic TBAs in order to enable controller synthesis. However, often it will be far easier and significantly less error-prone to express objectives in a suitable temporal logic, e.g. $\text{MTL}_{0,\infty}$.

Definition 2.5. An $\text{MTL}_{0,\infty}$ formula φ over actions Σ is defined by the grammar

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

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

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

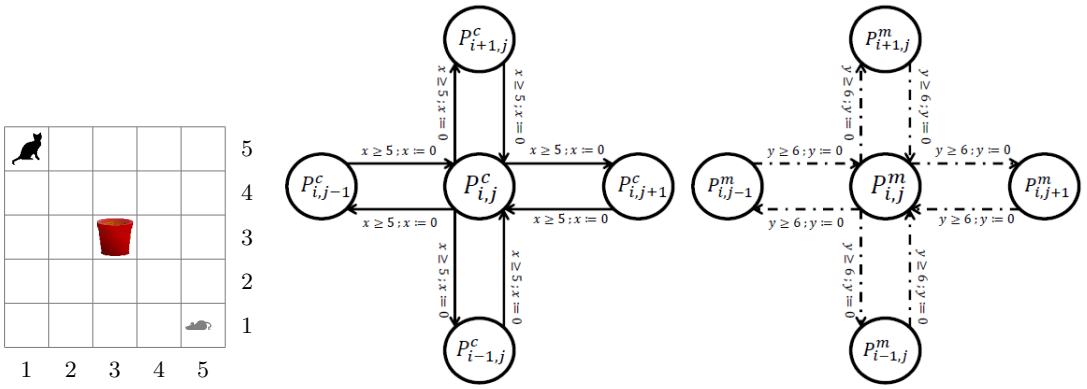
The semantics of $\text{MTL}_{0,\infty}$ is defined over *infinite* timed words. Let w^i be the i-th suffix of the timed word w. For a given infinite timed word $w = (t_1, a_1)(t_2, a_2)(t_3, a_3) \dots$ and an $\text{MTL}_{0,\infty}$ formula φ, the satisfaction relation $w^i \models \varphi$ is defined 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) $w^i \models \varphi_1 \mathsf{U}_{\sim d}\, \varphi_2$ 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) $w^i \models \varphi_1 \widehat{\mathsf{U}}_{\sim d}\, \varphi_2$ 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 $\text{MTL}_{0,\infty}$-formula φ iff $w^1 \models \varphi$. The language $\mathcal{L}(\varphi)$ of φ is the set of all infinite non-Zeno timed words that satisfy φ.

In [12], Doyen et al. proved that the controller synthesis problem for ECL (Event Clock logic) is undecidable. It is trival to check that all the future temporal operator in ECL can be defined in $\text{MTL}_{0,\infty}$(for instance, $\rhd_{[a,b]}\varphi$ can be defined as $(\widehat{\Box}_{<a}\, \neg\varphi) \wedge \widehat{\Diamond}_{\leq b}\varphi$. So the future fragment of ECL is a subset of $\text{MTL}_{0,\infty}$. In [12] some past time temporal operators, e.g. $\ominus$(the last-time) and $\lhd_{=0}$ (the last occurrence), are used to encode the configurations and the infinite space-bounded runs for lossy 3-counter machines. We find that these past time formulas can be replaced by some future time formulas: for instance, $\Box(\mathcal{Q} \rightarrow (\ominus tick \wedge \lhd_{=0} tick))$ can be replaced by $\Box(\bigcirc\mathcal{Q} \rightarrow (tick \wedge \rhd_{=0}\mathcal{Q}))$, and $\Box(c \rightarrow (\lhd_{=0}\mathcal{AB}))$ can be replaced by $\Box(\bigcirc c \rightarrow (\mathcal{AB} \wedge \lhd_{=0} c))$. Thus the controller synthesis problem for future fregment of ECL is also undecidable, and so is $\text{MTL}_{0,\infty}$. We summarise the above reasoning with the following theorem.

THEOREM 2.6. *The $\text{MTL}_{0,\infty}$ control synthesis problem is undecidable.*



(a) The *Arena* (b) The *Cat* (c) The *Mouse*

Figure 1: The *Arena* for Chasing Game (a) and snippets for the TG components for the *Cat* (b) and for the *Mouse* (c). Edges between P^c nodes are controllable (full) and labeled with the guard $x \geq 5$ and the reset $x := 0$. Edges between P^m nodes are uncontrollable (dashed) and labeled with the guard $y \geq 6$ and the reset $y := 0$.

Still, interesting properties exists for which we want to synthesise controllers.

Example 2.7. Reconsidering Example 2.4, we may formulate the first objective as $\Diamond Catch$, where $Catch = \vee_{i,j}(P^c_{i,j} \wedge P^m_{i,j})$. Repeated, and timed-bounded repeated catching may be expressed as the formulas $\Box\Diamond Catch$ and $\Box\Diamond_{\leq 40} Catch$. Finally, we may conjoin the formula $\Box\Diamond_{\geq 200} P^c_{1,5}$, which expresses that the *Cat* always will revisit its initial position after at least 200 time-units.

In the following sections we present a procedure for translating $\text{MTL}_{0,\infty}$ into a Timed Büchi Automaton. The translation goes by first translating the $\text{MTL}_{0,\infty}$ formula into a Transition-based Timed Büchi automata (TTBA) and subsequently using the degeneralization algorithm proposed in [15] to translate the TTBA into an equivalent TBA.

Definition 2.8. A Transition-based Timed Büchi automaton (TTBA) over actions Σ is a tuple $\mathcal{A} = (L, l_0, X, F, E)$, where L is the set of locations, l_0 is the initial location, X is a finite set of clocks, F is a finite set of accepting conditions, and $E \subseteq L \times \Sigma \times \Theta(X) \times 2^F \times 2^X \times L$ is the set of edges.

The set of states (including initial state $s_0 = (l_0, \overline{0})$) and the set of delay transitions of a TTBA are defined as for a TBA. For a TTBA there exists a *discrete* transition $(l_1, v_1) \xrightarrow{a,F_1} (l_2, v_2)$ if there exists an edge (l_1, a, g, F_1, Y, l_2) in the TTBA such that $v_1 \models g$ and $v_2 = v_1[Y]$.

A run of a TTBA is an infinite sequence of alternating delay and discrete transitions $s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{a_1,F_1} s_2 \xrightarrow{\delta_2} s_3 \xrightarrow{a_2,F_2} s_4 \xrightarrow{\delta_3} \dots$.

A timed word $w = (t_1, a_1)(t_2, a_2)(t_3, a_3)\dots$ over Σ is accepted by a TTBA $\mathcal{A}$ iff there exists states $s_0, s_1, s_2, \dots$ where s_0 is the initial state of $\mathcal{A}$ such that $s_0 \xrightarrow{t_1} s_1 \xrightarrow{a_1,F_1} s_2 \xrightarrow{t_2-t_1} s_3 \xrightarrow{a_2,F_2} s_4 \xrightarrow{t_3-t_2} \dots$ is a run of $\mathcal{A}$, and for each $f \in F$, there are infinitely many i where $f \in F_i$. We denote by $\mathcal{L}(\mathcal{A})$ the set of all non-Zeno timed words that are accepted by $\mathcal{A}$.

3 FROM $\text{MTL}_{0,\infty}$ TO TIMED BÜCHI AUTOMATA

In this section, we first present the translation of an $\text{MTL}_{0,\infty}$ into a TTBA, the translation goes through four phases. First we construct a closure for the formula in Section 3.1, giving the information needed for constructing extended formula. In Section 3.2 we continue by constructing extended formula containing book-keeping information for the time-constrained operators – such as monitoring clocks (and their resets). Next, we show how to transform such a formula (via a normal-form over the formula) into a TTBA. Finally one can derive deterministic over- and underapproximations for such a TTBA, based on the classical subset-construction from NFA to DFA – this construction is only subtly different than the one presented by Bulychev [8], we thus refrain from repeating it.

In the rest of this section, we assume that φ is an $\text{MTL}_{0,\infty}$ 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). Without loss og generality, we also assume that all temporal operators occurring in φ are included in $\{\mathsf{U}_{\leq d}, \mathsf{R}_{\leq d}, \mathsf{U}_{\geq d}, \mathsf{R}_{\geq d}\}$.

3.1 Closures & Extended Formulas

We use $\mathsf{Sub}(\varphi)$ to denote all the sub-formulas of φ. For each $\varphi_1 \mathsf{U}_{\leq d} \varphi_2 \in \mathsf{Sub}(\varphi)$, we assign a clock $x_{(\varphi_1 \mathsf{U}_{\leq d} \varphi_2)}$. Intuitively these clocks are used by the resulting TTBA to determine the time progression since starting to evaluate whether $(\varphi_1 \mathsf{U}_{\leq d} \varphi_2)$ is satisfied. We let $X_{U\leq} = \{x_{(\varphi_1 \mathsf{U}_{\leq d} \varphi_2)} \mid \varphi_1 \mathsf{U}_{\leq d} \varphi_2 \in \mathsf{Sub}(\varphi)\}$ be the set of all $\mathsf{U}_{\leq d}$-clocks and let $X_{U\geq}$, $X_{R\leq}$ and $X_{R\geq}$ be sets of clocks defined in a similar way. For

untimed modalities, $\varphi_1 \mathsf{U} \varphi_2$ and $\varphi_1 \mathsf{R} \varphi_2$, we do not assign clocks and thus assume $d > 0$ when we write $\mathsf{U}_{\geq d}$ or $\mathsf{R}_{\geq d}$ in this section.

The set of *basic formulas* for φ, written as $\mathsf{BF}(\varphi)$, is a finite set defined by the following rules:

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

Informally, $\varphi_1 \mathsf{U}_{\leq d-x} \varphi_2$ encodes that the TTBA has started evaluating $\varphi_1 \mathsf{U}_{\leq d} \varphi_2$ in a previous state (s) and therefore from the current state (s') the forumla $\varphi_1 \mathsf{U}_{\leq d'} \varphi_2$ should be satisfied where $d' = d - v(x)$ and $v(x)$ is the distance in time between s and s'. Similarly interpretation exists for $\mathsf{U}_{\leq d-x}$, $\mathsf{U}_{\leq d-x}$, $\mathsf{R}_{\leq d-x}$ and $\mathsf{R}_{\geq d-x}$. A formal definition is in Definition 3.1

As a conjunction of formulas can be represented as a set of formulas, we will use $2^{\mathsf{BF}(\varphi)}$ for both the powerset of $\mathsf{BF}(\varphi)$ and the set of all conjunctive formulas over $\mathsf{BF}(\varphi)$. Notice that because a conjunction with zero conjuncts is true then $true \in 2^{\mathsf{BF}(\varphi)}$. The closure of φ, denoted $\mathsf{CL}(\varphi)$, is the set of all positive Boolean combinations (i.e., without negation) over $\mathsf{BF}(\varphi)$. $\mathsf{CL}(\varphi)$ will form the set of non-initial locations for the deterministic TTBAs we construct for φ. Obviously, $\mathsf{CL}(\varphi)$ has only finitely many different non-equivalent formulas.

As information preserved in the closure and the basic formulas is not sufficient for the construction of the TTBA, we here introduce the notion of extended formula. Initially, for a given clock x we define the function $\mathsf{rst}(x)$ (and $\mathsf{unch}(x)$) for assigning clock-resets (and non-resets) of the clocks that track the temporal progress of the timed operators $\mathsf{U}_{\geq}$, $\mathsf{R}_{\leq}$ (and $\mathsf{U}_{\leq}$,$\mathsf{R}_{\geq}$). These functions are later used for constructing of the TTBA and capture "For the validity of the formula, when starting to evaluate a time-constrained operator $\mathsf{U}_{\geq}$ or $\mathsf{R}_{\leq}$ ($\mathsf{U}_{\leq}$ or $\mathsf{R}_{\geq}$), if $\mathsf{rst}(x)$ ($\mathsf{unch}(x)$) then x must be (must not be) reset". We here note that $\mathsf{rst}(x)$ only if $x \in X_{\mathsf{U}_{\geq}} \cup X_{\mathsf{R}_{\leq}}$, and symmetrically $\mathsf{unch}(x)$ only if $x \in X_{\mathsf{U}_{\leq}} \cup X_{\mathsf{R}_{\geq}}$. On can observe the application of rst (unch) in the definition of the function β in the rules 11 and 13 (10 and 16).

Let $F_\varphi = \{\psi \in \mathsf{Sub}(\varphi) |$ if there exists ψ_1 such that $\psi_1 \mathsf{U} \psi \in \mathsf{Sub}(\varphi)$ or $\psi_1 \mathsf{U}_{\geq d} \psi \in \mathsf{Sub}(\varphi)$ for some $d\}$. For each $\psi \in F_\varphi$ we introduce a boolean variable a_ψ that indicate that ψ is assumed to be false at the present state and define $\{a_\psi \mid \psi \in F_\varphi\}$ as the set of all such boolean variables for subformulas of φ. We shall later use F_φ to construct the acceptance condition for the TTBA in Section 3.2.

Now we define $\mathsf{Ext}(\varphi)$, the set of extended formulas for φ, with the following rules:

(1) $\mathsf{Sub}(\varphi) \subseteq \mathsf{Ext}(\varphi)$
(2) $\{a_\psi \mid \psi \in F_\varphi\} \subseteq \mathsf{Ext}(\varphi)$
(3) If $\phi \in \mathsf{CL}(\varphi)$, then $\phi, \bigcirc\phi \in \mathsf{Ext}(\varphi)$
(4) If $x \in X_{\mathsf{U}_{\leq}}$ or $x \in X_{\mathsf{R}_{\geq}}$, then $\mathsf{unch}(x) \in \mathrm{Ext}(\varphi)$
(5) If $x \in X_{\mathsf{U}_{\geq}}$ or $x \in X_{\mathsf{R}_{\leq}}$, then $\mathsf{rst}(x) \in \mathsf{Ext}(\varphi)$
(6) If $\Phi_1, \Phi_2 \in \mathsf{Ext}(\varphi)$, then $\Phi_1 \wedge \Phi_2$, $\Phi_1 \vee \Phi_2 \in \mathsf{Ext}(\varphi)$

$\mathsf{Ext}(\varphi)$ includes all the formulas needed to construct a TTBA for φ. Extended formulas can be interpreted using 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 all $x \in X$, 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 3.1. 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) $w^i \models_e a_\psi$ iff $\omega^i \not\models \psi$
(3) $\omega^i \models_e \mathsf{rst}(x)$ iff $v_{i+1}(x) = t_{i+1} - t_i$
(4) $\omega^i \models_e \mathsf{unch}(x)$ iff $v_{i+1}(x) = v_i(x) + t_{i+1} - t_i$
(5) $\omega^i \models_e \phi$ iff $w^i \models \phi$, if $\phi \in \mathsf{Sub}(\varphi)$
(6) $\omega^i \models_e \varphi_1 \mathsf{U}_{\sim d-x} \varphi_2$ 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 $\sim \in \{\leq, \geq\}$.
(7) $\omega^i \models_e \varphi_1 \mathsf{R}_{\sim d-x} \varphi_2$ 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 $\sim \in \{\leq, \geq\}$.
(8) $\omega^i \models_e \Phi_1 \wedge \Phi_2$ iff $\omega^i \models_e \Phi_1$ and $\omega^i \models_e \Phi_2$
(9) $\omega^i \models_e \Phi_1 \vee \Phi_2$ iff $\omega^i \models_e \Phi_1$ or $\omega^i \models_e \Phi_2$
(10) $\omega^i \models_e \bigcirc\Phi$ iff $\omega^{i+1} \models_e \Phi$

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

3.2 Constructing non-deterministic automata

Let us now construct a TTBA $\mathbb{A}_\varphi = (L, l_0, X, F, E)$ for which $\mathcal{L}(\mathbb{A}_\varphi) = \mathcal{L}(\varphi)$. The intuition of the elements of $\mathbb{A}$ is

- $L = \{\varphi\} \cup 2^{\mathsf{BF}(\varphi)}$, indicating that "in location $\ell \in L$ the future must satisfy ℓ",
- $\ell_0 = \varphi$, as the entire proposition is initially assumed satisfied,
- $X = X_{\mathsf{U}_{\geq}} \cup X_{\mathsf{R}_{\leq}} \cup X_{\mathsf{U}_{\leq}} \cup X_{\mathsf{R}_{\geq}}$ is the set of monitoring clocks,
- $F = F_\varphi$ is the set of accepting locations which must be visited infinitly often, and
- $E = (L \times \Sigma \times \Theta(X) \times 2^F \times 2^X \times L)$ is the transition relation where a single edge $(\psi, \alpha, g, F', \lambda, \psi')$ captures the inductive argument "when α is observed, ψ is true only if ψ' is true in the future given that g is satisfied and the clocks in λ are reset – implying that formulas in the set F' are false".

Edges of the TTBA are found by rewriting each $\psi \in \{\varphi\} \cup 2^{\mathsf{BF}(\varphi)}$ into a formula of $\mathsf{Ext}(\psi)$ that tells what action should be performed by the next transition, what clocks should be reset and what are the future obligations. This is similar to our work in [8] but in the current paper the rewriting also tells what subset of F_φ – which are assumed to be false. This difference is crucial since [8] did not consider Büchi acceptance conditions. The rewriting is done by the β function, capturing the condition for the input formula to be satisfied at the "current point in time" while using the next-operator to specify "under what condition is the next observation valid, what should be satisfied after the next observation, and what monitoring clocks should be reset" – a intuition is utilized when construction a normal-form over the rewritten formula. We inductively define β as

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

As an example, let us briefly discuss rules 9 and 10. Here the transformation of rule 9 states that either φ_2 is true already or after the next observation φ_1 must be true, the temporal constraint $x \leq d$ must be respected, and ϕ_1 must be true until ϕ_2 under the restriction that d is deducted by the amount monitored by x. The transformation of rule 10 is similar to the above, however, we also require that the monitoring clock x is not reset as the clock is vital for tracking the validity of the entire formula. The remaining rules 12-16 follow a similar pattern.

From the rules defining β, we note that the rules are constructed in such a way that alternative futures are separated by disjunction and each alternative is "guarded" by a necessary condition. For instance, let $\varphi = \alpha_1 \mathsf{U} \alpha_2$ for $\alpha_1, \alpha_2 \in \Sigma$, then for φ to be satisfied, either the next observation is α_2, and φ is satisfied, or the next observation is α_1, in which case, α_2 must not be observed – and the next observation must recursively satisfy φ again. As we will now generalize, this implies that our TTBA must have a transition from φ to φ, given that α_1 is observed and not α_2 – as well as a transition from φ to an accepting state under the condition that α_2 is observed.

From the definition of β we can see that $\beta(\psi)$ is an extended formula in $\mathsf{Ext}(\varphi)$. From the semantics given in Section 2.1 for $\mathrm{MTL}_{0,\infty}$, we know that $(\bigvee_{a \in \Sigma} a) \equiv \mathit{true}$ and for any $a, b \in \Sigma$, if $a \neq b$, then $a \wedge b \equiv \mathit{false}$. 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:

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

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

We call each $a_j \wedge g_j \wedge A_j^a \wedge \mathsf{rst}(X_j) \wedge \mathsf{unch}(Y_j) \wedge \bigcirc(\psi_j)$ a basic conjunction of $\beta(\psi)$. From each basic conjunction $a_j \wedge g_j \wedge A_j^a \wedge \mathsf{rst}(X_j) \wedge \mathsf{unch}(Y_j) \wedge \bigcirc(\psi_j)$ of $\beta(\psi)$ we define the transitions from ψ to ψ_j by

$$(\psi, a_j, g_j, F_j, \lambda, \psi_j) \in E \quad \textit{iff} \quad F_j = F \backslash A_j \textit{ and } X_j \subseteq \lambda \subseteq (X \backslash Y_j)$$

THEOREM 3.2. *Let φ be an $MTL_{0,\infty}$formula over Σ, and let $\mathbb{A}_\varphi$ be the TTBA built according to the procedure given above. Then $\mathcal{L}(\mathbb{A}_\varphi) = \mathcal{L}(\varphi)$.*

Given a basic conjunction $a \wedge g \wedge A^a \wedge \mathit{rst}(X_1) \wedge \mathsf{unch}(Y_1) \wedge \bigcirc(\psi_1)$, its sub-formula $\mathsf{rst}(X_1) \wedge \mathsf{unch}(Y_1)$ 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 what to do with the remaining clocks. In the construction so far we thus enumerate all the possible situations for clocks in $X \setminus (X_1 \cup Y_1)$, and hence get a non-deterministic choice as to what clocks to reset for a basic conjunction. However, we will see that this particular non-determinism can be avoided as there exists a best choice, which is to reset all clocks in $(X_{\mathsf{U}\leq} \cup X_{\mathsf{R}\geq}) \setminus Y_1$ and keep all clocks in $(X_{\mathsf{U}\geq} \cup X_{\mathsf{R}\leq}) \setminus X_1$ unchanged. The intuition of this choice is that each clock $x \in (X_{\mathsf{U}\leq} \cup X_{\mathsf{R}\geq})$ should be reset to zero unless $\mathsf{unch}(x)$ is asked to be true, and each clock $x \in (X_{\mathsf{U}\geq} \cup X_{\mathsf{R}\leq})$ should not be reset unless $\mathsf{rst}(x)$ is asked to be true. Using this approach, for a given basic conjunction $a \wedge g \wedge A^a \wedge \mathit{rst}(X_1) \wedge \mathsf{unch}(Y_1) \wedge \bigcirc(\psi_1)$ of $\beta(\psi)$, the transition $(\psi, a, g, F \setminus A, \lambda, \psi_1)$ with $\lambda = (X_1 \cup ((X_{U\leq} \cup X_{R\geq}) \setminus Y_1))$ will be the unique transition from ψ to ψ_1.

THEOREM 3.3. *Let φ be a $MTL_{0,\infty}$ formula over Σ and $\mathcal{A}$ be the an TTBA with best-choice-clock-resets above , then $\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}(\mathbb{A}_\varphi)$.*

Example 3.4. Let $\Sigma = \{p, !p\}$ and $f = \Box(p \rightarrow \bigcirc(!p \,\mathsf{U}_{\geq 10}\, p))$, then $X = X_{\mathsf{U}\geq} = \{x\}$, $X_{\mathsf{U}\leq} = X_{\mathsf{R}\geq} = X_{\mathsf{R}\leq} = \emptyset$, $F = \{p\}$, and

$$\begin{aligned}\beta(f) &= \beta(p \rightarrow \bigcirc(!p \mathsf{U}_{\geq 10} p)) \wedge \bigcirc(f) \\ &= (!p \vee \bigcirc(!p \mathsf{U}_{\geq 10} p)) \wedge \bigcirc(f) \\ &= (!p \wedge \bigcirc(f)) \vee (p \wedge \bigcirc(f \wedge f_1)) \vee (!p \wedge \bigcirc(f \wedge f_1)),\end{aligned}$$

where $f_1 = !p \,\mathsf{U}_{\geq 10}\, p$.

By the construction of $\mathbb{A}_f$, f will has 6 outgoing transitions: $(f, !p, \text{true}, \{\}, \{p\}, f)$, $(f, !p, \text{true}, \{x\}, \{p\}, f)$, $(f, p, \text{true}, \{\}, \{p\}, f \wedge f_1)$, $(f, p, \text{true}, \{x\}, \{p\}, f \wedge f_1)$, $(f, !p, \text{true}, \{\}, \{p\}, f \wedge f_1)$, $(f, !p, \text{true}, \{x\}, \{p\}, f \wedge f_1)$.

By theorem 3.3, the following three can be removed: $(f, !p, \text{true}, \{x\}, \{p\}, f)$, $(f, p, \text{true}, \{x\}, \{p\}, f \wedge f_1)$, $(f, !p, \text{true}, \{x\}, \{p\}, f \wedge f_1)$. The other three will remain. Similarly we can compute the outgoing transitions for $f \wedge f_1$, etc.

We observe that the structure of the disjunctive normal form gives the sufficient conditions for generating the under and over-approximations by applying the method discussed by Bulychev [8]. Fig.2(a) shows us the reduced TTBA $\mathcal{A}_f$ and Fig.2(b) shows us the determinized under-approximation.

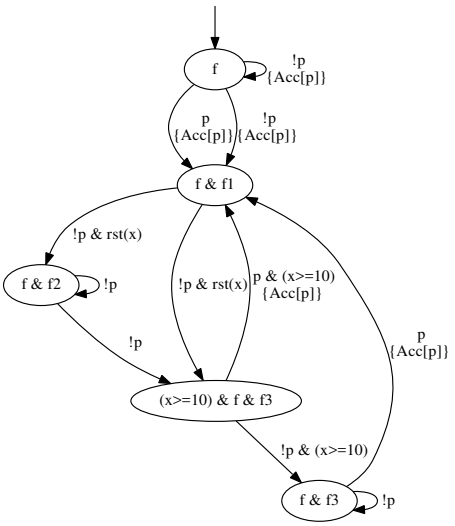
4 TOOL CHAIN

The conversions of $\text{MTL}_{0,\infty}$ to TBA has been implemented in the tool component CASAAL- a stand alone tool that was first described in [8]. In Figure 3 we illustrate the work flow of using CASAAL in combination with UPPAAL-TIGA. The starting-point of the workflow is a standard UPPAAL-TIGA TG $\mathcal{G}$ together with an $\text{MTL}_{0,\infty}$ property ϕ. The TG $\mathcal{G}$ is manually instrumented into $\mathcal{G}'$ to make the propositions in ϕ visible for the constructed monitor. CASAAL then takes ϕ and constructs under- and over-approximating deterministic TBA $\mathcal{A}^u$ and $\mathcal{A}^o$. Furthermore, CASAAL constructs a new combined TGB $\mathcal{G}' \oplus \mathcal{A}^u$ that is then passed on to UPPAAL-TIGA that will attempt to construct a winning strategy for the given property.

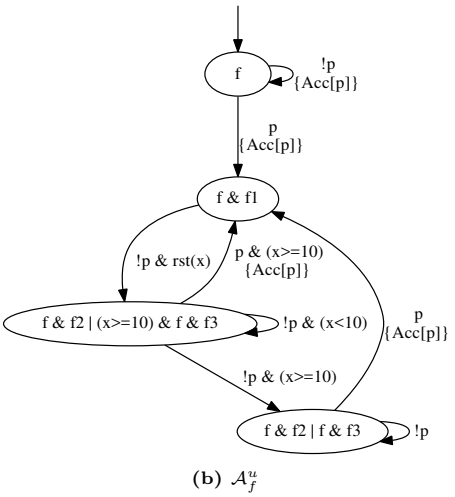
4.1 Experimental Evaluation

Cat and Mouse Example. As our first evaluation example we consider the *Cat* and *Mouse* game from Example 2.4. As objectives we consider the properties in Table 1. The properties are choosen to span the expressive power of $\text{MTL}_{0,\infty}$ covering both safety, liveness and mixtures. Table 1 reports for each property the size of the generated TBA in terms of number of edges and locations. It furthermore reports the time and memory consumed by CASAAL for constructing the TBA. For UPPAAL-TIGA we measure the time and memory used for synthesising a strategy. Finally the number of zones making up the strategy is reported as a means of quantifying the size of the strategy. We note that for all except one of the considered properties the tool chain provides exact answers as to whether a strategy exists or not[2]. We also note that for the last property, the TBA constructed for the under-approximation had no accepting states, making it a tautology.

[2]All of the experiments reported were done using CASAAL version 3.0 and a development snapshot of UPPAAL-TIGA running on an Intel Core i7-4578U 3.0 GHz processor. UPPAAL-TIGA was run using the following options: `--search-order 0 --backwards-order 0 --priority-order 2`. Both tools were limited to 4GB of memory and individually allowed to compute for up to 2 hours.



(a) $\mathcal{A}_f$



(b) $\mathcal{A}^u_f$

Figure 2: The resulting automata for $f = \Box(p \rightarrow \bigcirc(!p \,\mathsf{U}_{\geq 10}\, p))$**, where** $f_1 = !p \,\mathsf{U}_{\geq 10}\, p$**,** $f_2 = !p \,\mathsf{U}_{\geq 10-x}\, p$**, and** $f_3 = !p \,\mathsf{U}\, p$**.**

Train-Gate Example. As our second example we consider the classic and scalable UPPAAL Train-Gate example used for illustrating verification using UPPAAL. Here the challenge is to automatically synthesise correct-by-construction control strategies with respect to various objectives using our tool chain. In the example a number of trains has to pass over a common bridge, while the control strategy to be synthesised will take different actions to ensure safety and a variation of (bounded) liveness objectives.

A train is initially in Safe location and may approach (uncontrollably) at any moment. When the train is approaching it will alert the controller, which in turn should be synthezised to take appropriate actions to ensure the

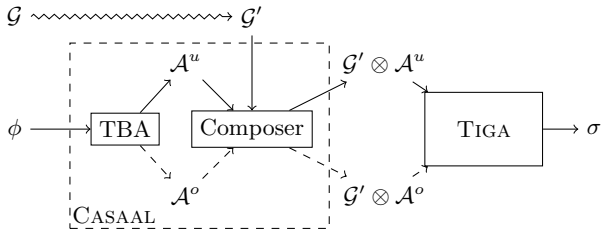


Figure 3: The tool chain workflow. The squiggly arrow indicate a manually performed step. The dashed arrow indicated the symmetric flow for the over approximation.

CASAAL				UPPAAL-TIGA			
		$\|L\|$	$\|E\|$	$\exists\sigma$	Time	Mem	#Zones
-	ψ_1	5	15	*false*	1.10	19	N/A
-	ψ_2	4	12	*true*	21.69	25	11417
-	ψ_3	3	9	*true*	32.38	27	13022
-	ψ_4	5	18	*true*	35.77	29	12996
-	ψ_5	9	45	*false*	130.90	45	N/A
-	ψ_6	9	45	*true*	151.30	45	28851
-	ψ_7	9	45	*true*	150.44	44	28958
-	ψ_8	4	15	*true*	27.78	24	16676
-	ψ_9	3	9	*true*	4.17	27	8813
-	ψ_{10}	4	15	*false*	61.78	39	N/A
-	ψ_{11}	4	15	*true*	49.15	37	18380
U	ψ_{12}	7	32	*true*	11.79	50	111277
O	ψ_{12}	8	38	*true*	1.08	34	30812
U	ψ_{13}	6	43	*false*	0.00	0	N/A
O	ψ_{13}	8	61	*true*	1.09	32	24427

Table 1: Experimental results for the *Cat* and *Mouse* model from Example 1. Time is given in seconds and memory in megabytes. We omit the resource consumption of Casaal as these are negligible for the given formulas. Formulas marked with a dash yield equivalent TBAs for the under- and over-approximation, while U and O signify the under- and over-approximation respectively. Formulas, with description, can be found in Table 4.

objective. In particular, while approaching a train can only be stopped within 10 time units of its signal of approach. Once stopped, a train may at any point be restarted and granted access to the bridge by the controller – all depending on the given objective. We attempted to synthesise strategies for the controller for various parameterized formulas (see Table 2). The full results can be found In Table 5 in the appendix where we report the size of generated TBA, memory and time consumption of TBA generation, strategy existence, time/memory consumption and strategy size if such a strategy exists.

For the properties proposed in Table 2, we observe that the under- and over-approximation yield the same TBA, hence

ϕ_0	No collisions and all trains will cross after approaching $(\Box\neg \mathit{collision}) \wedge \bigwedge_{i=1}^{n}(\Box(\mathit{Appr}[i] \rightarrow \Diamond \mathit{Cross}[i]))$
ϕ_1	No collisions and all trains will cross after 10 time units after approaching. $(\Box\neg \mathit{collision}) \wedge \bigwedge_{i=1}^{n}((\Box(\mathit{Appr}[i] \rightarrow \Diamond_{\geq 10} \mathit{Cross}[i])))$
ϕ_2	No collisions and all trains will cross before 10 ten time units after approaching. $(\Box\neg \mathit{collision}) \wedge \bigwedge_{i=1}^{n}((\Box(\mathit{Appr}[i] \rightarrow \Diamond_{\leq 10} \mathit{Cross}[i])))$
ϕ_3	No collisions and all trains will cross before 30 time units after approaching. $(\Box\neg \mathit{collision}) \wedge \bigwedge_{i=1}^{n}((\Box(\mathit{Appr}[i] \rightarrow \Diamond_{\leq 30} \mathit{Cross}[i])))$
ϕ_4	No collisions and all trains will cross before 50 time units after approaching. $(\Box\neg \mathit{collision}) \wedge \bigwedge_{i=1}^{n}((\Box(\mathit{Appr}[i] \rightarrow \Diamond_{\leq 50} \mathit{Cross}[i])))$

Table 2: Specifications for the Train-Gate example.

the approximation is exact. We note that for all but ϕ_1 the formulas are tractable for CASAAL with a running time of less than a minute. Still, it is interesting to see how the size of the TBA increases quite rapidly for ϕ_1 when adding an extra train. For ϕ_2, ϕ_3 and ϕ_4, even though the generated TBA are equivalent in size, the time for synthesis used by UPPAAL-TIGA differs quite a lot. We here observe that ϕ_2 through ϕ_4 are structurally the same formula and differ only in the bounds provided. Thus the TGBs constructed by CASAAL will also be structurally equivalent – however, for UPPAAL-TIGA the difference in the provided bounds, and hence the clock-guards of the constructed TGB, will result in different intersections of zones. As a result, we can observe an increasing zone-fragmentation from ϕ_2 through ϕ_4.

Chinese Juggler. In our third case-study, we consider the synthesis-problem for the scalable Chinese Juggler. The juggler is tasked with keeping n plates balancing on sticks. If a plate has been balancing for more than s time units, it can turn unstable. If a disk is unstable, after u time-units it can fall to the ground and shatter. At the same time, the juggler can only stabilize the disks at a certain pace, leaving him to choose which plates to stabilize for achieving his goal. A classical control problem is to ensure that no disk breaks. no disk breaks. We instantiate the disks with $s_1 = 8, s_2 = 8, s_3 = 20, s_4 = 20$ and $u_1 = 6, u_2 = 3, u_3 = 10, u_4 = 3$ for the 0-4 and syntehsise controllers for the properties shown in Table 3 for $1 \leq n \leq 4$.

We observe that for all the propositions for the Chinese Juggler produce the constructed TTBA is exact. We also note that the resource-consumption of CASAAL is negligible. For the untimed proposition ϕ_0 we can see that the constructed TTBA are small, resulting in good scalability. However, ϕ_2 and ϕ_3 in particular, show and exponential growth in the number of transitions, leading to an explosion of the number of zones needed to represent the strategy, ultimately leading to poor performance for the largest instance.

ϕ_0	If a plate turns unstable, it eventually becomes stable $\bigwedge_{i=1}^{n}(\Box(unstable[i] \rightarrow \Diamond stable[i]))$
ϕ_1	If a plate turns unstable, it becomes stable within 5 time-units $\bigwedge_{i=1}^{n}(\Box(unstable[i] \rightarrow \Diamond_{\leq 5} stable[i]))$
ϕ_2	If a plate turns unstable, it becomes stable after 5 time-units $\bigwedge_{i=1}^{n}(\Box(unstable[i] \rightarrow \Diamond_{\geq 5} stable[i]))$

Table 3: Specifications for the Chinese-Juggler example.

The full results can be found In Table 6 in the appendix where we report the size of generated TBA, strategy existence, time/memory consumption and strategy size if such a strategy exists.

5 CONCLUSIONS

In this paper we have significantly extended the practical scope of automatic controller synthesis for real-time systems. In particular, our method supports synthesis for all objectives expressed in $\text{MTL}_{0,\infty}$, a sublogic of MTL containing LTL and rich enough to express a wide variety of safety, liveness and bounded liveness properties. In general the synthesis problem for $\text{MTL}_{0,\infty}$ is undecidable. We overcome this obstacle by a new algorithm implemented Casaal converting $\text{MTL}_{0,\infty}$ into under-approximating Timed Büchi Automata. Combined with Uppaal-Tiga supporting synthesis for Timed Games with Büchi conditions we obtain a complete tool chain. In our experimental evaluation we demonstrated that for complex $\text{MTL}_{0,\infty}$ we predominantly obtain "exact and tight" approximations, supporting our initial claim. Furthermore, we showed on a number of scalable examples that synthesis for $\text{MTL}_{0,\infty}$ objectives is feasible using our tool-chain.

Future work includes to refine our determinstic under- and over-approximation construction by using the breakpoint technique [22] for Büchi determinization.

REFERENCES

[1] R. Alur. Formal verification of hybrid systems. In *Proceedings of the ninth ACM international conference on Embedded software*, EMSOFT '11, pages 273–278, New York, NY, USA, 2011. ACM.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[3] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43:116–146, January 1996.

[4] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *Proceedings of the 19th International Conference on Computer Aided Verification*, number 4590 in LNCS, pages 121–125. Springer, 2007.

[5] P. Bouyer, L. Bozzelli, and F. Chevalier. Controller synthesis for MTL specifications. In *In Proc. 17th International Conference on Concurrency Theory (CONCUR'06*, 2006.

[6] T. Brihaye, M. Estiévenart, and G. Geeraerts. On mitl and alternating timed automata over infinite words. In *Formal Modeling and Analysis of Timed Systems*, pages 69–84. Springer, 2014.

[7] J. R. Buchi and L. H. Landweber. Solving Sequential Conditions by Finite-State Strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[8] P. Bulychev, A. David, K. G. Larsen, A. Legay, G. Li, D. B. Poulsen, and A. Stainer. Monitor-based statistical model checking for weighted metric temporal logic. In *LPAR*, 2012.

[9] P. Bulychev, A. David, K. G. Larsen, and G. Li. Efficient controller synthesis for a fragment of $\text{MTL}_{0,\infty}$. *Acta Informatica*, 51(3-4):165–192, 2014.

[10] A. Church. Logic, Arithmetic, Automata. In *Proc. International Mathematical Congress*, 1962.

[11] L. Doyen, G. Geeraerts, J. Raskin, and J. Reicher. 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*, pages 133–148. Springer, 2009.

[12] L. Doyen, G. Geeraerts, J.-F. Raskin, and J. Reichert. Realizability of real-time logics. In *Formal Modeling and Analysis of Timed Systems*, pages 133–148. Springer, 2009.

[13] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.*, 2(3):241–266, 1982.

[14] M. Geilen. An improved on-the-fly tableau construction for a real-time temporal logic. In *In International Conference on Computer Aided Verification*, pages 276–290. Springer, 2003.

[15] D. Giannakopoulou and F. Lerda. From states to transitions: Improving translation of ltl formulae to büchi automata. In *Formal Techniques for Networked and Distributed Sytems—FORTE 2002*, pages 308–326. Springer, 2002.

[16] O. Kupferman, N. Piterman, and M. Vardi. Safraless compositional synthesis. In *18th Conference on Computer Aided Verification*, pages 31–44, 2006.

[17] O. Kupferman and M. Y. Vardi. μ-calculus synthesis. In *MFCS*, pages 497–507, 2000.

[18] O. Maler, D. Nickovic, and A. Pnueli. From mitl to timed automata. In *FORMATS'06*, pages 274–289. Springer, 2006.

[19] O. Maler, D. Nickovic, and A. Pnueli. On synthesizing controllers from bounded-response properties. In *CAV*, pages 95–107, 2007.

[20] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *STACS*, pages 229–242, 1995.

[21] Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 6(1):68–93, Jan. 1984.

[22] A. Morgenstern, K. Schneider, and S. Lamberti. Generating deterministic ω-automata for most ltl formulas by the breakpoint construction. In *MBMV*, pages 119–128, 2008.

[23] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '89, pages 179–190, New York, NY, USA, 1989. ACM.

[24] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

ψ_1	Whenever the mouse is caught it, it should not caught again before after 10 time units and it should be caught infinitely often. $(\Box(Catch \rightarrow \bigcirc(\neg Catch \mathsf{U}_{\geq 10} Catch))) \wedge \Box \diamond Catch$
ψ_2	Catch the mouse within 100 time units $\diamond_{\leq 100} Catch$
ψ_3	Wheneever the cat is at its initial position; then the mouse is caught within 100 time units $\Box(Initial \implies \diamond_{\leq 100} Catch)$
ψ_4	Cat should be at its initial place within 10 time units and whenever its at initial position, it should catch the mouse within 100 time units. $\diamond_{\leq 10} \wedge \psi_3$
ψ_5	Cat should be at its initial place after 10 time units and whenever its at initial position, it should catch the mouse within 100 time units. $\diamond_{\geq 10} \wedge \psi_3$
ψ_6	Cat should be at its initial place after 10 time units and whenever its at initial position, it should catch the mouse within 110 time units. $\diamond_{\geq 10} \wedge \Box(Initial \implies \diamond_{\leq 110} Catch)$
ψ_7	Cat should always return to its initial position before 200 time units; and always catch the mouse within 110 time units after visiting initial $\Box \diamond_{\leq 200} \wedge \Box(Initial \implies \diamond_{\leq 110} Catch)$
ψ_8	Cat should always return to its initial position; and always catch the mouse within 110 time units after visiting initial $\Box \diamond \wedge \Box(Initial \implies \diamond_{\leq 110} Catch)$
ψ_9	After catching the mouse, the cat should return to its initial state within 40 time units. $\Box(Catch \implies \diamond_{\leq 40} Initial)$
ψ_{10}	When the cat is at its initial position, it should catch the mouse within 100 units and after catching the mouse it should return to its initial position within 40 time units. $(\Box Initial \implies \diamond_{\leq 100} Catch) \wedge (\Box(Catch) \implies \diamond_{\leq 40} Initial)$
ψ_{11}	When the cat is at its initial position, it should catch the mouse within 110 units and after catching the mouse it should return to its initial position within 40 time units. $(\Box Initial \implies \diamond_{\leq 110} Catch) \wedge (\Box(Catch) \implies \diamond_{\leq 40} Initial)$
ψ_{12}	Eventually, within 200 time units, the lazy cat moves away from the initial position and plays with the mouse for 50 time units. Within 100 units of moving from the initial position, the cat moves back to the initial position. $\diamond_{\geq 200}(\neg Initial \wedge (\Box_{\leq 50} \neg Catch \rightarrow Catch) \wedge (\diamond_{\leq 100} Initial))$
ψ_{13}	It always holds that within 200 time units, the lazy cat moves away from the initial position and plays with the mouse for 50 time units. Within 100 units of moving from the initial position, the cat moves back to the initial position. $\Box(\diamond_{\geq 200}(\neg Initial \wedge (\Box_{\leq 50} \neg Catch \rightarrow Catch)) \wedge (\diamond_{\leq 100} Initial))$

Table 4: Specifications for the *Cat* and *Mouse* example.

	Casaal					Uppaal-Tiga			
	#Trains	Time	Mem	#Loc	#Edges	$\exists\sigma$	Time	Mem	#Zones
ϕ_0	1	0.04	35	3	6	*true*	0.01	9	11
ϕ_1	1	0.04	35	5	17	*true*	0.01	9	21
ϕ_2	1	0.04	35	3	8	*true*	0.02	9	6
ϕ_3	1	0.04	35	3	8	*true*	0.02	9	12
ϕ_4	1	0.04	35	3	8	*true*	0.01	9	12
ϕ_0	2	0.04	35	7	30	*true*	0.01	9	152
ϕ_1	2	0.08	35	31	309	*true*	0.15	12	804
ϕ_2	2	0.03	35	5	40	*false*	0.09	10	N/A
ϕ_3	2	0.03	35	5	40	*false*	0.28	10	N/A
ϕ_4	2	0.03	35	5	40	*true*	0.20	10	132
ϕ_0	3	0.04	35	17	112	*true*	0.52	15	1556
ϕ_1	3	0.72	67	174	3719	*true*	58.59	216	24211
ϕ_2	3	0.04	36	9	170	*false*	16.77	56	N/A
ϕ_3	3	0.04	36	9	170	*false*	44.69	77	N/A
ϕ_4	3	0.04	36	9	170	*true*	53.70	81	2738
ϕ_0	4	0.05	37	41	360	*true*	26.95	141	14479
ϕ_1	4	10.48	365	893	37256	??	TO		
ϕ_2	4	0.13	40	17	664	??	TO		
ϕ_3	4	0.08	40	17	664	??	TO		
ϕ_4	4	0.08	40	17	664	??	TO		
ϕ_0	5	0.17	40	97	1056	*true*	2740.56	3192	132371
ϕ_1	5	174.15	3175	4358	336505	??	TO		
ϕ_2	5	0.23	56	33	2462	??	TO		
ϕ_3	5	0.21	56	33	2462	??	TO		
ϕ_4	5	0.20	56	33	2462	??	TO		

Table 5: Experimental results for the Train-Gate controller synthesis. Time is given in seconds and memory in megabytes.

	Casaal			Uppaal-Tiga			
	n	$\|L\|$	$\|E\|$	$\exists\sigma$	Time	Mem	#Zones
ψ_1	1	2	4	*true*	0.01	10	11
ψ_2	1	3	6	*true*	0.01	10	13
ψ_3	1	4	13	*true*	0.01	10	22
ψ_1	2	6	24	*true*	0.03	10	119
ψ_2	2	5	36	*true*	0.26	10	179
ψ_3	2	30	279	*true*	2.24	10	903

	Casaal			Uppaal-Tiga			
	n	$\|L\|$	$\|E\|$	$\exists\sigma$	Time	Mem	#Zones
ψ_1	3	16	96	*true*	4.26	18	1498
ψ_2	3	9	162	*true*	504.31	116	2997
ψ_3	3	173	3546	??	TO		
ψ_1	4	40	320	*true*	790.78	333	26300
ψ_2	4	17	648	??	TO		
ψ_3	4	892	36364	??	TO		

Table 6: Experimental results for the Chinese-Juggler controller synthesis. Time is given in seconds, memory in megabytes and n gives the number of plates being juggled.

A PROOFS FOR THEOREM 3.2

In this Appendix we provide a proof for Theorem 3.2. The following two lemmas proves the validity of the rewrite rules.

LEMMA A.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. f 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 $\omega \models_e \varphi_1 U_{\leq d}\, \varphi_2 f$.

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

The proofs for remaining cases are similar to the above and are omitted. □

Definition A.2. 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 $\varphi_1 \mathsf{U}_{\leq d}\, \varphi_2 \in \mathrm{Sub}(\varphi)$, then

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

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

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

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

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

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

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

LEMMA A.3. *Let w be a timed word, and $\overline{w}$ be the extended timed word defined in Definition A.2, 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 A.4. *For each $\psi \in Sub(\varphi) \cup CL(\varphi)$, $\beta(\psi)$ can be transformed equivalently into a disjunction of basic conjunctions:*

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

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

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(ψ_1)=0, if ψ_1 is an action or a clock bound;
(4). Length($\bigcirc\psi_1$)=0;
(5). Length(ψ_1 *op* ψ_2)= Length(ψ_1)+ Length(ψ_2)+1, where *op* is an operator in the set $\{\wedge, \vee, \mathsf{U}, \mathsf{U}_{\leq d}, \mathsf{U}_{\geq d}, \mathsf{U}_{\leq d-x}, \mathsf{U}_{\geq d-x}, \mathsf{R}, \mathsf{R}_{\leq d}, \mathsf{R}_{\geq d}, \mathsf{R}_{\leq d-x}, \mathsf{R}_{\geq d-x}\}$.

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

$$\beta(\varphi_1) \equiv \bigvee_{i=1}^{k_1} \left(a_i' \wedge g_i' \wedge A_i'^a \wedge rst(X_i') \wedge unch(Y_i') \wedge \bigcirc(\psi_i')\right)$$

and

$$\beta(\varphi_2) \equiv \bigvee_{j=1}^{k_2} \left(a_j'' \wedge g_j'' \wedge A_j''^a \wedge rst(X_j'') \wedge unch(Y_j'') \wedge \bigcirc(\psi_j'')\right),$$

then

$$\beta(\psi) \equiv \bigvee_{i=1}^{k_1} \bigvee_{j=1}^{k_2} \left((a_i' \wedge a_j'') \wedge (g_i' \wedge g_j'') \wedge (A_i' \cup A_j'')^a \wedge rst(X_i' \cup X_j'') \wedge unch(Y_i' \cup Y_j'')\right.$$

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) Remaining cases are similar to the above. □

Theorem 3.2. Let φ be an $\mathrm{MTL}_{0,\infty}$formula over Σ, and $\mathbb{A}_\varphi$ be a transition-based timed Büchi automaton for φ built according to the procedure given above. Then $\mathcal{L}(\mathbb{A}_\varphi) = \mathcal{L}(\varphi)$.

PROOF. 1. $\mathcal{L}(\mathbb{A}_\varphi) \subseteq \mathcal{L}(\varphi)$.

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 $\psi_1, \psi_2, \psi_3, \ldots \in L$, $(\psi_1, a_1, g_1, F_1, r_1, \psi_2)$, $(\psi_2, a_2, g_2, F_2, r_2, \psi_3), \ldots \in \mathbb{E}$ 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 $X_i \subseteq (X_{\mathsf{U}\geq} \cup X_{\mathsf{R}\leq})$ and $Y_i \subseteq (X_{\mathsf{U}\leq} \cup X_{\mathsf{R}\geq})$ such that $a_i \wedge g_i \wedge A_i^a \wedge \mathit{rst}(X_i) \wedge \mathit{unch}(Y_i) \wedge \bigcirc(\psi_{i+1})$ is a basic conjunction of $\beta(\psi_i)$, $X_i \subseteq r_i \subseteq (X \setminus Y_i)$, $v_i \models g_i$, $A_i = F \setminus F_i$ 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 $\omega^i \models_e a_i \wedge g_i \wedge \mathit{rst}(X_i) \wedge \mathit{unch}(Y_i)$.

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

(1) For each $a \in \Sigma$, $\mu_i(a) = \mathit{true}$ iff $a = a_i$
(2) $\mu_i(x \sim d) = \mathit{true}$ iff $v_i(x) \sim d$
(3) For each $f \in F$, $\mu_i(a_f) = \mathit{true}$ iff $f \in A_i$
(4) $\mu_i(\mathit{rst}(x)) = \mathit{true}$ iff $v_{i+1}(x) = t_{i+1} - t_i$
(5) $\mu_i(\mathit{unch}(x)) = \mathit{true}$ iff $v_{i+1}(x) = v_i(x) + t_{i+1} - t_i$
(6) For each $\varphi_1 \in \mathrm{BF}(\varphi)$, $\mu_i(\bigcirc(\varphi_1)) = \mathit{true}$ iff $\varphi_1 \in \psi_{i+1}$ (please noted that ψ_{i+1} is a subset of $\mathrm{BF}(\varphi)$)
(7) $\mu_i(\varphi_1 \wedge \varphi_2) = \mu_i(\varphi_1) \wedge \mu_i(\varphi_2)$
(8) $\mu_i(\varphi_1 \vee \varphi_2) = \mu_i(\varphi_1) \vee \mu_i(\varphi_2)$
(9) $\mu_i(\varphi_1 \,\mathsf{U}\, \varphi_2) = \mu_i(\varphi_2) \vee (\mu_i(\varphi_1) \wedge \mu_i(a_{\varphi_2}) \wedge \mu_i(X(\varphi_1 \,\mathsf{U}\, \varphi_2)))$
(10) $\mu_i(\varphi_1 \,\mathsf{R}\, \varphi_2) = \mu_i(\varphi_2) \wedge (\mu_i(\varphi_1) \vee \mu_i(\bigcirc(\varphi_1 \,\mathsf{R}\, \varphi_2)))$
(11) $\mu_i(\varphi_1 \,\mathsf{R}_{\leq d}\, \varphi_2) = \mu_i(\varphi_2) \wedge (\mu_i(\varphi_1) \vee \mu_i(\mathrm{rst}(x)) \wedge (\mu_i(\bigcirc(\varphi_1 \,\mathsf{R}_{\leq d-x} \varphi_2)) \vee \mu_i(\bigcirc(x > d))))$, where x is the clock assigned to $\varphi_1 \,\mathsf{R}_{\leq d}\, \varphi_2$
(12) The other cases for $\mu_i(\varphi_1 \,\mathsf{U}_{\leq d}\, \varphi_2)$, $\mu_i(\varphi_1 \,\mathsf{U}_{\leq d-x}\, \varphi_2)$, $\mu_i(\varphi_1 \,\mathsf{R}_{\leq d-x}\, \varphi_2)$, $\mu_i(\varphi_1 \,\mathsf{U}_{\geq d}\, \varphi_2)$, $\mu_i(\varphi_1 \,\mathsf{U}_{\geq d-x}\, \varphi_2)$, $\mu_i(\varphi_1 \,\mathsf{R}_{\geq d-x}\, \varphi_2)$ can be defined similarly, according to the rewriting rules in Section 3.2.

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

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

Now we show that for each $\psi \in \mathrm{Sub}(\varphi) \cup \mathrm{CL}(\varphi)$, if $\mu_i(\psi) = \mathit{true}$, 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 $\mu_i(\psi) = \mathit{true}$,
then $\phi \in \psi_{i+1}$, and $\mu_{i+1}(\phi) = \mathit{true}$.
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 $\mu_i(\psi) = \mathit{true}$,
then $\mu_i(\phi_1) = \mu_i(\phi_2) = \mathit{true}$.
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 $\mu_i(\psi) = \mathit{true}$,
then $\mu_i(\phi_1) = \mathit{true}$ or $\mu_i(\phi_2) = \mathit{true}$.
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 $\psi = \phi_1 \,\mathsf{U}\, \phi_2$ and $\mu_i(\psi) = \mathit{true}$,
then $\mu_i(\varphi_2) \vee (\mu_i(\varphi_1) \wedge \mu_i(a_{\varphi_2}) \wedge \mu_i(\bigcirc(\varphi_1 \,\mathsf{U}\, \varphi_2))) = \mathit{true}$.

Thus we get that $\mu_i(\varphi_2) = \mathit{true}$, or that $\mu_i(\varphi_1) = \mathit{true}$, $\varphi_2 \in A_i$ and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+1}$.

By induction, we get that $\omega^i \models \varphi_2$ or that ($\omega^i \models \varphi_1$, $\varphi_2 \in A_i$, and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+1}$).

Case 1. If $\omega^i \models \varphi_2$, then $\omega^i \models \phi_1 \mathsf{U} \phi_2$, and the conclusion is true.

Case 2. If $\omega^i \models \varphi_1$, $\varphi_2 \in A_i$ and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+1}$, then from $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+1}$ we know that $\mu_{i+1}(\varphi_1 \,\mathsf{U}\, \varphi_2) = \mathit{true}$, Thus we get that $\mu_{i+1}(\varphi_2) = \mathit{true}$, or that $\mu_{i+1}(\varphi_1) = \mathit{true}$, $\varphi_2 \in A_{i+1}$ and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+2}$.

By induction, we get that $\omega^{i+1} \models \varphi_2$ or that ($\omega^{i+1} \models \varphi_1$, $\varphi_2 \in A_{i+1}$, and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+2}$).

(2.1). If $\omega^{i+1} \models \varphi_2$, then $\omega^{i+1} \models \phi_1 \mathsf{U} \phi_2$. Since we alright know that $\omega^i \models \varphi_1$, thus $\omega^i \models \phi_1 \mathsf{U} \phi_2$, and so the conclusion is true.

(2.2). If $\omega^{i+1} \models \varphi_1$, $\varphi_2 \in A_{i+1}$, and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+2}$, then from $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+2}$ we know that $\mu_{i+2}(\varphi_1 \,\mathsf{U}\, \varphi_2) = \mathit{true}$, Thus we get that
$\mu_{i+2}(\varphi_2) = \mathit{true}$, or that $\mu_{i+2}(\varphi_1) = \mathit{true}$, $\varphi_2 \in A_{i+2}$ and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+3}$.
By induction, we get that $\omega^{i+2} \models \varphi_2$ or that $\omega^{i+2} \models \varphi_1$, $\varphi_2 \in A_{i+2}$, and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in \psi_{i+3}$.
.....

This procedure will eventually stop at least when $\varphi_2 \in F_j$ for some $j \geq i$. Thus we get that $\omega^i \models \phi_1 \,\mathsf{U}\, \phi_2$.

(6) If $\psi = \phi_1 \mathsf{R} \phi_2$ and $\mu_i(\psi) = \mathit{true}$,
then $\mu_i(\varphi_2) \wedge (\mu_i(\varphi_1) \vee \mu_i(\bigcirc(\varphi_1 \,\mathsf{R}\, \varphi_2)))$ is true.
Thus we get that $\mu_i(\varphi_2) = \mathit{true}$, and that $\mu_i(\varphi_1) = \mathit{true}$ or $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+1}$.
By induction, we get that $\omega^i \models_e \varphi_2$, $\omega^i \models_e \varphi_1$ or $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+1}$.

Case 1. If $\omega^i \models_e \varphi_2$ and $\omega^i \models_e \varphi_1$, then $\omega^i \models_e \phi_1 \mathsf{R} \phi_2$, and the conclusion is true.

Case 2. If $\omega^i \models_e \varphi_2$ and $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+1}$, then from $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+1}$ we know that $\mu_{i+1}(\varphi_1 \,\mathsf{R}\, \varphi_2) = \mathit{true}$, thus $\mu_{i+1}(\varphi_2) = \mathit{true}$, and $\mu_{i+1}(\varphi_1) = \mathit{true}$ or $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+2}$.
By induction, we have that $\omega^{i+1} \models_e \varphi_2$, $\omega^{i+1} \models_e \varphi_1$ or $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+2}$.

(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 $\omega^i \models_e \phi_1 \mathsf{R} \phi_2$. So the conclusion is true.

(2.2). If $\omega^{i+1} \models_e \varphi_2$ and $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+2}$, then we know that
$\mu_{i+2}(\varphi_1 \,\mathsf{R}\, \varphi_2) = \mathit{true}$.
Thus $\mu_{i+2}(\varphi_2) = \mathit{true}$, and $\mu_{i+2}(\varphi_1) = \mathit{true}$ or $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+3}$.
By induction, we get that $\omega^{i+2} \models_e \varphi_2$, $\omega^{i+2} \models_e \varphi_1$ or $\varphi_1 \,\mathsf{R}\, \varphi_2 \in \psi_{i+3}$.

This procedure can stop or proceed infinitely; in both case, we could get that $\omega^i \models_e \phi_1 \mathsf{R} \phi_2$.

(7) If $\psi = \phi_1 \,\mathsf{U}_{\geq d}\, \phi_2$ and $\mu_i(\psi) = \mathit{true}$,

then $\mu_i(\varphi_1) \wedge (\mu_i(\varphi_2) \vee \mu_i(a_{\varphi_2})) \wedge \mu_i(rst(x)) \wedge \mu_i(\bigcirc((\varphi_1 \mathsf{U}_{\geq d-x} \varphi_2) \vee ((x \geq d) \wedge (\varphi_1 \mathsf{U} \varphi_2)))) = true$.

Thus, we get that $\omega^i \models_e \varphi_1$, $v_{i+1}(x) = (t_{i+1} - t_i)$, $\varphi_1 \mathsf{U}_{\geq d-x} \varphi_2 \in \psi_{i+1}$ or $(x \geq d) \wedge (\varphi_1 \mathsf{U} \varphi_2) \in \psi_{i+1}$.

If $(x \geq d) \wedge (\varphi_1 \mathsf{U} \varphi_2) \in \psi_{i+1}$, then $t_{i+1} - t_i \geq d$, $\omega^{i+1} \models_e \varphi_1 \mathsf{U} \varphi_2$, and we get the conclusion that $\omega^i \models_e \varphi_1 \mathsf{U}_{\geq d} \varphi_2$.

If $\varphi_1 \mathsf{U}_{\geq d-x} \varphi_2 \in \psi_{i+1}$, then we get that $\mu_{i+1}(\varphi_1) \wedge (\mu_{i+1}(\varphi_2) \vee \mu_{i+1}(a_{\varphi_2})) \wedge \mu_{i+1}(\bigcirc((\varphi_1 \mathsf{U}_{\geq d-x} \varphi_2) \vee ((x \geq d) \wedge (\varphi_1 \mathsf{U} \varphi_2)))) = true$.

So $\omega^{i+1} \models_e \varphi_1$, $\varphi_1 \mathsf{U}_{\geq d-x} \varphi_2 \in \psi_{i+2}$ or $(x \geq d) \wedge (\varphi_1 \mathsf{U} \varphi_2) \in \psi_{i+2}$.

If $(x \geq d) \wedge (\varphi_1 \mathsf{U} \varphi_2) \in \psi_{i+2}$, we will get the conclusion that $\omega^i \models_e \varphi_1 \mathsf{U}_{\geq d} \varphi_2$.

If $\varphi_1 \mathsf{U}_{\geq d-x} \varphi_2 \in \psi_{i+2}$, we continue the above procedure until $t_{i+k} - t_i \geq d$ and $\varphi_2 \in \psi_{i+k}$ for some $k > 2$. This can be guaranteed by the fact that $\varphi_2 \in F_{i+k}$ will become true for infinitely many times.

(8) Remaining cases are omitted as they are quite similar to the above.

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

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

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 A.2.

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

Let $\varphi_1 = \varphi$, then by Lemma A.3, 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 $\alpha_1 \wedge g_1 \wedge A_1^a \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\varphi_2)$ of φ_1 such that $\overline{w} \models_e \alpha_1 \wedge g_1 \wedge A_1^a \wedge rst(X_1) \wedge unch(Y_1) \wedge \bigcirc(\varphi_2)$.

Thus $\alpha_1 = a_1$, $v_1 \models g_1$, $\overline{w} \models_e A_1^a$, $\overline{w} \models_e rst(X_1) \wedge unch(Y_1)$, and $\overline{w}^2 \models_e \varphi_2$.

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

Let $F_1 = F \setminus A_1$, then from the construction in Section 3.2, $(\varphi_1, a_1, g_1, F_1, r_1,$
$\varphi_2) \in \mathbb{E}$ 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 A.3, we know that there is a basic conjunction $\alpha_2 \wedge g_2 \wedge A_2^a \wedge rst(X_2) \wedge unch(Y_2) \wedge \bigcirc(\varphi_3)$ of φ_2 such that $\overline{w}^2 \models_e \alpha_2 \wedge g_2 \wedge A_2^a \wedge rst(X_2) \wedge unch(Y_2) \wedge \bigcirc(\varphi_3)$.

Thus $\alpha_2 = a_2$, $v_2 \models g_2$, $\overline{w}^2 \models_e A_2^a$, $\overline{w}^2 \models_e rst(X_2) \wedge unch(Y_2)$, and $\overline{w}^3 \models_e \varphi_3$.

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

Let $F_2 = F \setminus A_2$, then $(\varphi_2, a_2, g_2, F_2, r_2, \varphi_3) \in \mathbb{E}$ 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 an infinite sequence $\rho = (\varphi_1, a_1, g_1,$
$F_1, r_1, \varphi_2)(\varphi_2, a_2, g_2, F_2, r_2, \varphi_3)(\varphi_3, a_3, g_3, F_3, r_3, \varphi_4)\ldots$, which is a run of $\mathbb{A}_\varphi$ over w.

Now it suffices to prove that ρ is an accepting run. On the contrary, if we assume that ρ is not an accepting run, then there exists a $f \in F$ and a $n \in \mathbb{N}$ such that $f \notin F_k$ for all $k > n$. Since $F_k = F \setminus A_k$ and $\overline{w}^k \models_e A_k^a$, we know that $f \in A_k$ and $w^k \not\models f$ for all $k > n$.

From $f \in A_k$, we can also knew that there exists some $g\mathsf{U}f \in \mathrm{Sub}(\varphi)$(or $g\mathsf{U}_{\geq d} f \in \mathrm{Sub}(\varphi)$) such that $g\mathsf{U}f \in \psi_{k+1}$ (or $g\mathsf{U}_{\geq d-x} f \in \psi_{k+1}$). Since $\overline{w}^{k+1} \models_e \psi_{k+1}$, we then get that $\overline{w}^{k+1} \models_e g\mathsf{U}f$ or $\overline{w}^{k+1} \models_e g\mathsf{U}_{\geq d-x} f$, thus there exists $j \geq k+1$ such that $\overline{w}^j \models f$. This contrasts with the conclusion that $w^k \not\models f$ for all $k > n$. So ρ is an accepting run of $\mathbb{A}_\varphi$ and $w \in \mathcal{L}(\mathbb{A}_\varphi)$.

□

□

B PROOF FOR THEOREM 3.3

Theorem 3.3. Let φ be a $\mathrm{MTL}_{0,\infty}$ formula over Σ and $\mathcal{A}$ be the an TTBA with best-choice-clock-resets above , then $\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}(\mathbb{A}_\varphi)$.

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

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 $\psi_1, \psi_2, \psi_3, \ldots \in L$, $(\psi_1, a_1, g_1, F_1, r_1, \psi_2)$, $(\psi_2, a_2, g_2, F_2, r_2, \psi_3), \ldots \in \mathbb{E}$ and $v_1, v_2, v_3, \ldots \in \mathbb{R}_{\geq 0}^X$ such that $\psi_1 = \varphi$, and for each $i \geq 1$: there are $X_i \subseteq X_{\mathsf{U}\geq} \cup X_{\mathsf{R}\leq}$ and $Y_i \subseteq X_{\mathsf{U}\leq} \cup X_{\mathsf{R}\geq}$ such that $a_i \wedge g_i \wedge (F \setminus F_i)^a \wedge rst(X_i) \wedge unch(Y_i) \wedge \bigcirc(\psi_{i+1})$ is a basic conjunction of $\beta(\psi_i)$, $X_i \subseteq r_i \subseteq X \setminus Y_i$, $v_i \models g_i$, and $v_{i+1} = (v_i[r_i]) + (t_{i+1} - t_i)$.

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

We define $v_1', v_2', v_3', \ldots \in \mathbb{R}_{\geq 0}^X$ 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 is true.

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

From the rewrite rules in Section 3.2 , it is easy to see that the following Assertion 2 is also true.

Assertion 2: For each $x \in X$, if $x \in X_{\mathsf{U}\geq} \cup X_{\mathsf{R}\leq}$, then $x > d$ and $x \geq d$ will not occur in g_i; if $x \in X_{\mathsf{U}\leq} \cup X_{\mathsf{R}\geq}$, then $x < d$ 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 $(\psi_1, \overline{0}) \xrightarrow{t_1} (\psi_1, v_1') \xrightarrow{a_1, F_1} (\psi_2, v_1'[\lambda_1]) \xrightarrow{t_2 - t_1} (\psi_2, v_2') \xrightarrow{a_2, F_2} (\psi_3, v_2'[\lambda_2]) \xrightarrow{t_3 - t_2} (\psi_3, v_3')\ldots$ 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.

□