Model-based Verification and Analysis for Real-Time Systems

Uli Fahrenberg, Kim Guldstrand Larsen, Claus Thrane

To cite this version:


HAL Id: hal-01088054
https://inria.hal.science/hal-01088054
Submitted on 27 Nov 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Model-based Verification and Analysis
for Real-Time Systems

Uli Fahrenberg, Kim G. Larsen 1, Claus Thrane

Department of Computer Science, Aalborg University, Denmark

Abstract.
This article aims at providing a concise and precise Travellers Guide, Phrase Book or Reference Manual to the timed automata modeling formalism introduced by Alur and Dill [7, 8]. The paper gives comprehensive definitions of timed automata, priced (or weighted) timed automata, and timed games and highlights a number of results on associated decision problems related to model checking, equivalence checking, optimal scheduling, and the existence of winning strategies.

Keywords. Timed automata, regions, zones, reachability, bisimilarity; priced and weighted timed automata, optimal reachability, optimal infinite runs, conditional optimality; timed games, winning strategies.

1. Introduction

The model of timed automata, introduced by Alur and Dill [7, 8], has by now established itself as a classical formalism for describing the behaviour of real-time systems. A number of important algorithmic problems has been shown decidable for it, including reachability, model checking and several behavioural equivalences and preorders.

By now, real-time model checking tools such as UPPAAL [17, 59] and KRONOS [32] are based on the timed automata formalism and on the substantial body of research on this model that has been targeted towards transforming the early results into practically efficient algorithms — e.g. [13, 14, 19, 21] — and data structures — e.g. [20, 56, 58].

The maturity of a tool like UPPAAL is witnessed by the numerous applications — e.g. [38, 40, 47, 50, 54, 57, 62, 63] — to the verification of industrial case-studies spanning real-time controllers and real-time communication protocols. More recently, model-checking tools in general and UPPAAL in particular have been applied to solve realistic scheduling problems by a reformulation as reachability problems — e.g. [1, 44, 49, 64].

Aiming at providing methods for performance analysis, a recent extension of timed automata is that of priced or weighted timed automata [9, 18], which makes it possible to formulate and solve optimal scheduling problems. Surprisingly, a

1 Corresponding Author: Kim G. Larsen, Department of Computer Science, Aalborg University, Selma Lagerlöfs Vej 300, 9220 Aalborg Øst, Denmark. E-mail: e-mail: kgl@cs.aau.dk
number of properties have been shown to be decidable for this formalism [9,18,29,42,60]. The recently developed UPPAAL CORA tool provides an efficient tool for solving cost-optimal reachability problems [55] and has been applied successfully to a number of optimal scheduling problems, e.g. [15,22,46].

Most recently, substantial efforts have been made on the automatic synthesis of (correct-by-construction) controllers from timed games for given control objectives. From early decidability results [12,65] the effort has lead to efficient on-the-fly algorithms [34,71] with the newest of the UPPAAL toolset, UPPAAL TIGA [16], providing an efficient tool implementation with industrial applications emerging, e.g. [52].

This survey paper aims at providing a concise and precise Travellers Guide, Phrase Book or Reference Manual to the land and language of timed automata. The article gives comprehensive definitions of timed automata, weighted timed automata, and timed games and highlights a number of results on associated decision problems related to model checking, equivalence checking, optimal scheduling, and the existence of winning strategies. The intention is that the paper should provide an easy-to-access collection of important results and overview of the field to anyone interested.

The authors would like to thank the students of the Marktoberdorf and Quantitative Model Checking PhD schools for their useful comments and help in weed ing out a number of errors in the first edition of this survey [43], as well as an anonymous reviewer who provided many useful remarks for the invited paper [41] at FSEN 2009.

2. Timed automata

In this section we review the notion of timed automata introduced by Alur and Dill [7, 8] as a formalism for describing the behaviour of real-time systems. We review the syntax and semantics and highlight the, by now classical, region construction underlying the decidability of several associated problems.

Here we illustrate how regions are applied in showing decidability of reachability and timed and untimed (bi)similarity. However, the notion of region does not provide the means for efficient tool implementations. The verification engine of UPPAAL instead applies so-called zones, which are convex unions of regions. We give a brief account of zones as well as their efficient representation and manipulation using difference-bound matrices.

2.1. Syntax and semantics

**Definition 2.1.** The set $\Phi(C)$ of clock constraints $\varphi$ over a finite set (of clocks) $C$ is defined by the grammar

$$
\varphi ::= x \bowtie k \mid \varphi_1 \land \varphi_2 \quad (x \in C, k \in \mathbb{Z}, \bowtie \in \{\leq, <, \geq, >\}).
$$

The set $\Phi^+(C)$ of extended clock constraints $\varphi$ is defined by the grammar

$$
\varphi ::= x \bowtie k \mid x - y \bowtie k \mid \varphi_1 \land \varphi_2 \quad (x, y \in C, k \in \mathbb{Z}, \bowtie \in \{\leq, <, \geq, >\}).
$$
Remark 2.2. The clock constraints in $\Phi(C)$ above are also called diagonal-free clock constraints, and the additional ones in $\Phi^+(C)$ are called diagonal. We restrict ourselves to diagonal-free clock constraints here; see Remark 2.44 for one reason. For additional modelling power, timed automata with diagonal constraints can be used, as it is shown in [8, 26] that any such automaton can be converted to a diagonal-free one; however the conversion may lead to an exponential blow-up.

Definition 2.3. A timed automaton is a tuple $(L, \ell_0, F, C, \Sigma, I, E)$ consisting of a finite set $L$ of locations, an initial location $\ell_0 \in L$, a set $F \subseteq L$ of final locations, a finite set $C$ of clocks, a finite set $\Sigma$ of actions, a location invariants mapping $I : L \to \Phi(C)$, and a set $E \subseteq L \times \Phi(C) \times \Sigma \times 2^C \times L$ of edges.

Here $2^C$ denotes the set of subsets (i.e. the power set) of $C$. We shall write $\ell \xrightarrow{a,r} \ell'$ for an edge $(\ell, \varphi, a, r, \ell') \in E$. In figures, resets are written as assignment to zero, e.g. $x := 0$.

Example 2.1. Figure 1 provides a timed automaton model of an intelligent light switch. Starting in the “Off” state, a press of the button turns the light on, and it remains in this state for 100 time units (i.e. until clock $x = 100$), at which time the light turns off again. During this time, an additional press resets the clock $x$ and prolongs the time in the state by 100 time units. Pressing the button twice, with at most three time units between the presses, triggers a special bright light.

Definition 2.4. A clock valuation on a finite set $C$ of clocks is a mapping $v : C \to \mathbb{R}_{\geq 0}$. The initial valuation $v_0$ is given by $v_0(x) = 0$ for all $x \in C$. For a valuation $v$, $d \in \mathbb{R}_{\geq 0}$, and $r \subseteq C$, the valuations $v + d$ and $v[r]$ are defined by
\((v + d)(x) = v(x) + d\)
\[v[r](x) = \begin{cases} 0 & \text{for } x \in r, \\ v(x) & \text{for } x \notin r. \end{cases}\]

Extending the notation for power set introduced above, we will in general write \(B^A\) for the set of mappings from a set \(A\) to a set \(B\). The set of clock valuations on \(C\) is thus \(\mathbb{R}^{\leq 0}_C\).

**Definition 2.5.** The zone of an extended clock constraint in \(\Phi^+(C)\) is a set of clock valuations \(C \rightarrow \mathbb{R}_{\geq 0}\) given inductively by
\[
[x \bowtie k] = \{v : C \rightarrow \mathbb{R}_{\geq 0} \mid v(x) \bowtie k\},
\]
\[
[x - y \bowtie k] = \{v : C \rightarrow \mathbb{R}_{\geq 0} \mid v(x) - v(y) \bowtie k\}, \text{ and}
\]
\[
[\varphi_1 \land \varphi_2] = [\varphi_1] \cap [\varphi_2].
\]
We shall write \(v \models \varphi\) instead of \(v \in [\varphi]\).

**Definition 2.6.** The semantics of a timed automaton \(A = (L, \ell_0, F, C, \Sigma, I, E)\) is the transition system \([A] = (S, s_0, \Sigma \cup \mathbb{R}_{\geq 0}, T = T_s \cup T_d)\) given as follows:
\[
S = \{(\ell, v) \in L \times \mathbb{R}^{\leq 0}_C \mid v \models I(\ell)\} \quad s_0 = (\ell_0, v_0)
\]
\[
T_s = \{(\ell, v) \xrightarrow{a} (\ell', v') \mid \exists \ell' : v \models \varphi, v' = v[r]\}
\]
\[
T_d = \{(\ell, v) \xrightarrow{d} (\ell, v + d) \mid \forall d' \in [0, d] : v + d' \models I(\ell)\}
\]

**Remark 2.7.** The transition system \([A]\) from above is an example of what is known as a timed transition system, i.e. a transition system where the label set includes \(\mathbb{R}_{\geq 0}\) as a subset and which satisfies certain additivity and time determinacy properties. We refer to [2] for a more in-depth treatment.

Also note that the semantics \([A]\) contains no information about final states (derived from the final locations in \(F\)); this is mostly for notational convenience.

**Definition 2.8.** A (finite) run of a timed automaton \(A = (L, \ell_0, F, C, \Sigma, I, E)\) is a finite path \(\rho = (\ell_0, v_0) \rightarrow \cdots \rightarrow (\ell_k, v_k)\) in \([A]\). It is said to be accepting if \(\ell_k \in F\).

**Example 2.1** (continued). The light switch model from figure 1 has as state set
\[
S = \{\text{Off}\} \times \mathbb{R}_{\geq 0} \cup \{\text{Light, Bright}\} \times [0, 100]
\]
where we identify valuations with their values at \(x\). A few example runs are given below; we abbreviate “press?” to “p”:
\[
\begin{align*}
(\text{Off}, 0) & \xrightarrow{150} (\text{Off}, 150) \xrightarrow{p} (\text{Light, 0}) \xrightarrow{100} (\text{Light, 100}) \rightarrow (\text{Off}, 0) \\
(\text{Off}, 0) & \xrightarrow{p} (\text{Light, 0}) \xrightarrow{10} (\text{Light, 10}) \xrightarrow{p} (\text{Light, 0}) \xrightarrow{100} (\text{Light, 100}) \rightarrow (\text{Off}, 0) \\
(\text{Off}, 0) & \xrightarrow{p} (\text{Light, 0}) \xrightarrow{1} (\text{Light, 1}) \xrightarrow{p} (\text{Bright, 0}) \xrightarrow{100} (\text{Bright, 100}) \rightarrow (\text{Off}, 0)
\end{align*}
\]
2.2. Reachability

We are concerned with the following problem: Given a timed automaton $A = (L, \ell_0, F, C, \Sigma, I, E)$, is any of the locations in $F$ reachable? We shall later define the timed language generated by a timed automaton and see that this reachability problem is equivalent to emptiness checking: Is the timed language generated by $A$ non-empty?

Example 2.2 (cf. [2, Ex. 11.7]). Figure 2 shows a timed automaton $A$ with two clocks and a final location $\ell_1$. To ask whether $\ell_1$ is reachable amounts for this automaton to the question whether there is a finite sequence of $a$- and $b$-transitions from $\ell_0$ which brings clock values into accordance with the guard $x \geq 4 \land y \leq 2$ on the edge leading to $\ell_1$.

An immediate obstacle to reachability checking is the infinity of the state space of $A$. In general, the transition system $[A]$ has uncountably many states, hence straight-forward reachability algorithms do not work for us.

Notation 2.9. The derived transition relations in a timed automaton $A = (L, \ell_0, F, C, \Sigma, I, E)$ are defined as follows: For $((\ell, v), (\ell', v'))$ states in $[A]$, we say that

- $\delta(\ell, v) \xrightarrow{d} (\ell', v')$ if $((\ell, v)) \xrightarrow{d} (\ell', v')$ in $[A]$ for some $d > 0$,
- $\alpha(\ell, v) \xrightarrow{a} (\ell', v')$ if $((\ell, v)) \xrightarrow{a} (\ell', v')$ in $[A]$ for some $a \in \Sigma$, and
- $\ell, v \leadsto (\ell', v')$ if $((\ell, v)) (\xrightarrow{d} \cup \xrightarrow{a})^* (\ell', v')$.

Definition 2.10. The set of reachable locations in a timed automaton $A = (L, \ell_0, F, C, \Sigma, I, E)$ is

$$\text{Reach}(A) = \{ \ell \in L \mid \exists v : C \rightarrow \mathbb{R}_{\geq 0} : (\ell_0, v_0) \leadsto (\ell, v) \}.$$ 

Hence we can now state the reachability problem as follows:

Problem 2.1 (Reachability). Given a timed automaton $A = (L, \ell_0, F, C, \Sigma, I, E)$, is Reach$(A) \cap F \neq \emptyset$ ?
Definition 2.11. Let $A = (L, \ell_0, F, C, \Sigma, I, E)$ be a timed automaton. A relation $R \subseteq L \times \mathbb{R}_{\geq 0} \times L \times \mathbb{R}_{\geq 0}$ is a time-abstracted simulation provided that for all $(\ell_1, v_1) R (\ell_2, v_2)$,

- for all $(\ell_1, v_1) \xrightarrow{\delta} (\ell'_1, v'_1)$ there exists some $(\ell'_2, v'_2)$ such that $(\ell'_1, v'_1) R (\ell'_2, v'_2)$ and $(\ell_2, v_2) \xrightarrow{\delta} (\ell'_2, v'_2)$, and
- for all $a \in \Sigma$ and $(\ell_1, v_1) \xrightarrow{a} (\ell'_1, v'_1)$, there exists some $(\ell'_2, v'_2)$ such that $(\ell'_1, v'_1) R (\ell'_2, v'_2)$ and $(\ell_2, v_2) \xrightarrow{a} (\ell'_2, v'_2)$.

$R$ is said to be $F$-sensitive if additionally, $(\ell_1, v_1) R (\ell_2, v_2)$ implies that $\ell_1 \in F$ if and only if $\ell_2 \in F$. A time-abstracted bisimulation is a time-abstracted simulation which is also symmetric; we write $(\ell_1, v_1) \approx (\ell_2, v_2)$ whenever $(\ell_1, v_1) R (\ell_2, v_2)$ for a time-abstracted bisimulation $R$.

Note that $\approx$ is itself a time-abstracted bisimulation, which is easily shown to be an equivalence relation and hence symmetric, reflexive and transitive. Observe also that a time-abstracted (bi)simulation on $A$ is the same as a standard (bi)simulation on the transition system derived from $[A]$ with transitions $\xrightarrow{\delta}$ and $\xrightarrow{a}$. Likewise, the quotient introduced below is just the standard bisimulation quotient of this derived transition system.

Definition 2.12. Let $A = (L, \ell_0, F, C, \Sigma, I, E)$ be a timed automaton and $R \subseteq L \times \mathbb{R}_{\geq 0} \times L \times \mathbb{R}_{\geq 0}$ a time-abstracted bisimulation which is also an equivalence. The quotient of $[A] = (S, s_0, \Sigma \cup \{\delta\}, T_R)$ with respect to $R$ is the transition system $[A]_R = (S_R, s^0_R, \Sigma \cup \{\delta\}, T_R)$ given by $S_R = S/R, s^0_R = [s_0]_R$, and with transitions

- $\pi \xrightarrow{\delta} \pi'$ whenever $(\ell, v) \xrightarrow{\delta} (\ell', v')$ for some $(\ell, v) \in \pi$, $(\ell', v') \in \pi'$, and
- $\pi \xrightarrow{a} \pi'$ whenever $(\ell, v) \xrightarrow{a} (\ell', v')$ for some $(\ell, v) \in \pi$, $(\ell', v') \in \pi'$.

The following proposition expresses that $F$-sensitive quotients are sound and complete with respect to reachability.

Proposition 2.13 ([4]). Let $A = (L, \ell_0, F, C, \Sigma, I, E)$ be a timed automaton, $R \subseteq L \times \mathbb{R}_{\geq 0} \times L \times \mathbb{R}_{\geq 0}$ an $F$-sensitive time-abstracted bisimulation and $\ell \in F$. Then $\ell \in \text{Reach}(A)$ if and only if there is a reachable state $\pi$ in $[A]_R$ and $v : C \to \mathbb{R}_{\geq 0}$ such that $(\ell, v) \in \pi$.

Example 2.2 (continued). We shall now try to construct, in a naïve way, a time-abstracted bisimulation $R$ for the timed automaton $A$ from Figure 2 which is as coarse as possible. Note first that we cannot have $(\ell_0, v) R (\ell_1, v')$ for any $v, v' : C \to \mathbb{R}_{\geq 0}$ because $\ell_1 \in F$ and $\ell_0 \notin F$. On the other hand it is easy to see that we can let $(\ell_1, v) R (\ell_1, v')$ for all $v, v' : C \to \mathbb{R}_{\geq 0}$, which leaves us with constructing $R$ on the states involving $\ell_0$.

We handle switch transitions $\xrightarrow{a}$ first: If $v, v' : C \to \mathbb{R}_{\geq 0}$ are such that $v(y) \leq 2$ and $v'(y) > 2$, the state $(\ell_0, v)$ has an $a$-transition available while the state $(\ell_0, v')$ has not, hence these cannot be related in $R$. Similarly we have to distinguish states $(\ell_0, v)$ from states $(\ell_0, v')$ where $v(x) \leq 2$ and $v'(x) > 2$ because
of $b$-transitions, and states $(\ell_0, v)$ from states $(\ell_0, v')$ where $v(x) < 4$ and $v'(x) \geq 4$ because of $c$-transitions. Altogether this gives the five classes depicted to the left of Figure 3, where the shading indicates to which class the boundary belongs, and we have written the set of available actions in the classes.

When also taking delay transitions $\delta \mapsto$ into account, one has to partition the state space further: From a valuation $v$ in the class marked $\{a, b\}$ in the left of the figure, a valuation in the class marked $\{a\}$ can only be reached by a delay transition if $v(y) < v(x)$; likewise, from the $\{a\}$ class, the $\{a, c\}$ class can only be reached if $v(y) \leq v(x) - 2$. Hence these two classes need to be partitioned as shown to the right of Figure 3.

It can easily be shown that no further partitioning is needed, thus we have defined the coarsest time-abstracted bisimulation relation for $A$, altogether with eight equivalence classes.

### 2.3. Regions

Motivated by the construction in the example above, we now introduce a time-abstracted bisimulation with a finite quotient. To ensure finiteness, we need the maximal constants to which respective clocks are compared in the invariants and guards of a given timed automaton. These may be defined as follows.

**Definition 2.14.** For a finite set $C$ of clocks, the maximal constant mapping $c_{\text{max}} : C \rightarrow \mathbb{Z}^{\Phi(C)}$ is defined inductively as follows:

$$c_{\text{max}}(x)(y \triangleleft k) = \begin{cases} k & \text{if } y = x \\ 0 & \text{if } y \neq x \end{cases}$$

$$c_{\text{max}}(x)(\varphi_1 \land \varphi_2) = \max\left(c(x)(\varphi_1), c(x)(\varphi_2)\right)$$

For a timed automaton $A = (L, \ell_0, F, C, \Sigma, I, E)$, the maximal constant mapping is $c_A : C \rightarrow \mathbb{Z}$ defined by

$$c_A(x) = \max\{c_{\text{max}}(x)(I(\ell)), c_{\text{max}}(x)(\varphi) \mid \ell \in L, \ell \xrightarrow{c_{\text{max}}} \ell' \in E\}.$$
Figure 4. Clock regions for the timed automaton from Example 2.2.

Notation 2.15. For \( d \in \mathbb{R}_{\geq 0} \) we write \( \lfloor d \rfloor \) and \( \langle d \rangle \) for the integral, respectively fractional, part of \( d \), so that \( d = \lfloor d \rfloor + \langle d \rangle \).

Definition 2.16. For a timed automaton \( A = (L, \ell_0, F, C, \Sigma, I, E) \), valuations \( v, v' : C \to \mathbb{R}_{\geq 0} \) are said to be region equivalent, denoted \( v \equiv v' \), if

- \( \lfloor v(x) \rfloor = \lfloor v'(x) \rfloor \) or \( v(x), v'(x) > c_A(x) \), for all \( x \in C \), and
- \( \langle v(x) \rangle = 0 \) iff \( \langle v'(x) \rangle = 0 \), for all \( x \in C \), and
- \( \langle v(x) \rangle \leq \langle v(y) \rangle \) iff \( \langle v'(x) \rangle \leq \langle v'(y) \rangle \) for all \( x, y \in C \).

Proposition 2.17 ([4]). For a timed automaton \( A = (L, \ell_0, F, C, \Sigma, I, E) \), the equivalence relation \( \equiv \) defined on states of \([A]\) by \( (\ell, v) \equiv (\ell', v') \) if \( \ell = \ell' \) and \( v \equiv v' \) is an \( F \)-sensitive time-abstracted bisimulation. The quotient \([A]_{\equiv}\) is finite.

The equivalence classes of valuations of \( A \) with respect to \( \equiv \) are called regions, and the quotient \([A]_{\equiv}\) is called the region automaton associated with \( A \).

Proposition 2.18 ([8]). The number of regions for a timed automaton \( A \) with a set \( C \) of \( n \) clocks is bounded above by

\[
n! \cdot 2^n \cdot \prod_{x \in C} (2c_A(x) + 2) .
\]

Example 2.2 (continued). The 69 regions of the timed automaton \( A \) from Figure 2 are depicted in Figure 4.

Propositions 2.13 and 2.17 together now give the decidability part of the theorem below; for PSPACE-completeness see [6,37].

Theorem 2.19. The reachability problem for timed automata is PSPACE-complete.

2.4. Behavioural refinement relations

We have already introduced time-abstracted simulations and bisimulations in Definition 2.11. As a corollary of Proposition 2.17, these are decidable:
Theorem 2.20. Time-abstracted simulation and bisimulation are decidable for timed automata.

Proof: One only needs to see that time-abstracted (bi)simulation in the timed automaton is the same as ordinary (bi)simulation in the associated region automaton; indeed, any state in $[A]$ is untimed bisimilar to its image in $[A]_R$. The result follows by finiteness of the region automaton.

The following provides a time-sensitive variant of (bi)simulation.

Definition 2.21. Let $A = (L, \ell_0, F, C, \Sigma, I, E)$ be a timed automaton. A relation $R \subseteq L \times \mathbb{R}_{\geq 0} \times L \times \mathbb{R}_{\geq 0}$ is a timed simulation provided that for all $(\ell_1, v_1)$ $R$ $(\ell_2, v_2)$,

- for all $(\ell_1, v_1) \xrightarrow{d} (\ell_1', v_1')$, $d \in \mathbb{R}_{\geq 0}$, there exists some $(\ell_2', v_2')$ such that $(\ell_1', v_1') R (\ell_2', v_2')$ and $(\ell_2, v_2) \xrightarrow{d} (\ell_2', v_2')$, and
- for all $(\ell_1, v_1) \xrightarrow{a} (\ell_1', v_1')$, $a \in \Sigma$, there exists some $(\ell_2', v_2')$ such that $(\ell_1', v_1') R (\ell_2', v_2')$ and $(\ell_2, v_2) \xrightarrow{a} (\ell_2', v_2')$.

A timed bisimulation is a timed simulation which is also symmetric, and two states $(\ell_1, v_1), (\ell_2, v_2) \in [A]$ are said to be timed bisimilar, written $(\ell_1, v_1) \sim (\ell_2, v_2)$, if there exists a timed bisimulation $R$ for which $(\ell_1, v_1) R (\ell_2, v_2)$.

Note that $\sim$ is itself a timed bisimulation on $A$, which is easily shown to be an equivalence relation and hence transitive, reflexive and symmetric.

Definition 2.22. Two timed automata $A = (L^A, \ell_0^A, F^A, C^A, \Sigma^A, I^A, E^A)$ and $B = (L^B, \ell_0^B, F^B, C^B, \Sigma^B, I^B, E^B)$ are said to be timed bisimilar, denoted $A \sim B$, if $(\ell_0^A, v_0) \sim (\ell_0^B, v_0)$ in the disjoint-union transition system $[A] \sqcup [B]$.

Timed simulation of timed automata can be analogously defined. The following decidability result was established for parallel timed processes in [36]; below we give a version of the proof which has been adapted for timed automata.

Theorem 2.23. Timed similarity and bisimilarity are decidable for timed automata.

Before the proof, we need a few auxiliary definitions and lemmas. The first is a product of timed transition systems which synchronizes on time, but not on actions:

Definition 2.24. The independent product of the timed transition systems $[A] = (S^A, s_0^A, \Sigma^A \cup \mathbb{R}_{\geq 0}, T^A)$, $[B] = (S^B, s_0^B, \Sigma^B \cup \mathbb{R}_{\geq 0}, T^B)$ associated with timed automata $A, B$ is $[A] \times [B] = (S, s_0, \Sigma^A \cup \Sigma^B \cup \mathbb{R}_{\geq 0}, T)$ given by

$$
S = S^A \times S^B \\
T = \{(p, q) \xrightarrow{a} (p', q) \mid a \in \Sigma, p \xrightarrow{a} p' \in T^A\} \\
\cup \{(p, q) \xrightarrow{b} (p, q') \mid b \in \Sigma, q \xrightarrow{b} q' \in T^B\} \\
\cup \{(p, q) \xrightarrow{d} (p', q') \mid d \in \mathbb{R}_{\geq 0}, p \xrightarrow{d} p' \in T^A, q \xrightarrow{d} q' \in T^B\}
$$
We need to extend region equivalence $\cong$ to the independent product. Below, $\oplus$ denotes vector concatenation (direct sum); note that $(p_1, q_1) \cong (p_2, q_2)$ is not the same as $p_1 \cong p_2$ and $q_1 \cong q_2$, as fractional orderings $(x^A) \triangleq (x^B)$, for $x^A \in C^A$, $x^B \in C^B$, have to be accounted for in the former, but not in the latter. Hence $(p_1, q_1) \cong (p_2, q_2)$ implies $p_1 \cong p_2$ and $q_1 \cong q_2$, but not vice-versa.

**Definition 2.25.** For states $p_i = (\ell^{p_i}, v^{p_i})$ in $\llbracket A \rrbracket$ and $q_i = (\ell^{q_i}, v^{q_i})$ in $\llbracket B \rrbracket$ for $i = 1, 2$, we say that $(p_1, q_1) \cong (p_2, q_2)$ iff $\ell^{p_1} = \ell^{p_2} \land \ell^{q_1} = \ell^{q_2}$ and $v^{p_1} \oplus v^{q_1} \cong v^{p_2} \oplus v^{q_2}$.

Note that the number of states in $\llbracket A \times [B] \rrbracket_{\cong}$ is finite, with an upper bound given by Proposition 2.18. Next we define transitions in $\llbracket A \times [B] \rrbracket_{\cong}$.

**Notation 2.26.** Regions in $\llbracket A \times [B] \rrbracket_{\cong}$ will be denoted $X, X'$. The equivalence class of a pair $(p, q) \in \llbracket A \times [B] \rrbracket_{\cong}$ is denoted $[p, q]$.

**Definition 2.27.** For $X, X' \in \llbracket A \times [B] \rrbracket_{\cong}$, we say that

- $X \xrightarrow{a} X'$ for $a \in \Sigma$ if for all $(p, q) \in X$ there exists $(p', q') \in X'$ such that $(p, q) \xrightarrow{a} (p', q')$ in $\llbracket A \times [B] \rrbracket_{\cong}$,
- $X \xrightarrow{b} X'$ for $b \in \Sigma$ if for all $(p, q) \in X$ there exists $(p', q') \in X'$ such that $(p, q) \xrightarrow{b} (p', q')$ in $\llbracket A \times [B] \rrbracket_{\cong}$, and
- $X \xrightarrow{d} X'$ if for all $(p, q) \in X$ there exists $d \in \mathbb{R}_{\geq 0}$ and $(p', q') \in X'$ such that $(p, q) \xrightarrow{d} (p', q')$.

**Definition 2.28.** A subset $B \subseteq \llbracket A \times [B] \rrbracket_{\cong}$ is a **symbolic bisimulation** provided that for all $X \in B$,

- whenever $X \xrightarrow{a} X'$ for some $X' \in \llbracket A \times [B] \rrbracket_{\cong}$, then $X' \xrightarrow{a} X''$ for some $X'' \in B$,
- whenever $X \xrightarrow{b} X'$ for some $X' \in \llbracket A \times [B] \rrbracket_{\cong}$, then $X' \xrightarrow{b} X''$ for some $X'' \in B$, and
- whenever $X \xrightarrow{d} X'$ for some $X' \in \llbracket A \times [B] \rrbracket_{\cong}$, then $X' \in B$.

Note that it is decidable whether $\llbracket A \times [B] \rrbracket_{\cong}$ admits a symbolic bisimulation. The following proposition finishes the proof of Theorem 2.23.

**Proposition 2.29.** The quotient $\llbracket A \times [B] \rrbracket_{\cong}$ admits a symbolic bisimulation if and only if $A \sim B$.

**Proof (cf. [36]):** For a given symbolic bisimulation $B \subseteq \llbracket A \times [B] \rrbracket_{\cong}$, the set $R_B = \{(p, q) \mid [p, q] \in B\} \subseteq \llbracket A \times [B] \rrbracket$ is a timed bisimulation. For the other direction, one can construct a symbolic bisimulation from a timed bisimulation $R \subseteq \llbracket A \times [B] \rrbracket$ by $B_R = \{(p, q) \mid (p, q) \in R\}$. \qed
2.5. Language inclusion and equivalence

Similarly to the untimed setting, there is also a notion of language inclusion and equivalence for timed automata. We need to introduce the notion of timed trace first. Note that we restrict to finite timed traces here; similar results are available for infinite traces in timed automata with Büchi or Muller acceptance conditions, see [8].

Definition 2.30. A timed trace over a finite set of actions \( \Sigma \) is a finite sequence \( ((t_1, a_1), (t_2, a_2), \ldots, (t_k, a_k)) \), where \( a_i \in \Sigma \) and \( t_i \in \mathbb{R}_{\geq 0} \) for \( i = 1, \ldots, k \), and \( t_i < t_{i+1} \) for \( i = 1, \ldots, k - 1 \). The set of all timed traces over \( \Sigma \) is denoted \( T\Sigma^* \).

In a pair \((t_i, a_i)\), the number \( t_i \) is called the time stamp of the action \( a_i \), i.e. the time at which event \( a_i \) occurs.

Remark 2.31. Timed traces as defined above are also known as strongly monotonic timed traces, because of the assumption that no consecutive events occur at the same time. Weakly monotonic timed traces, i.e. with requirement \( t_i \leq t_{i+1} \) instead of \( t_i < t_{i+1} \), have also been considered, and there are some subtle differences between the two; see [67] for an important example.

Definition 2.32. A timed trace \( ((t_1, a_1), \ldots, (t_k, a_k)) \) is accepted by a timed automaton \( A = (L, \ell_0, F, C, \Sigma, I, E) \) if there is an accepting run

\[
(\ell_0, v_0) \xrightarrow{t_1} (\ell_0, v_0 + t_1) \xrightarrow{a_1} (\ell_1, v_1) \xrightarrow{t_2-t_1} \cdots \xrightarrow{a_k} (\ell_k, v_k)
\]

in \( A \). The timed language of \( A \) is \( L(A) = \{ \tau \in T\Sigma^* | \tau \text{ accepted by } A \} \).

It is clear that \( L(A) = \emptyset \) if and only if none of the locations in \( F \) is reachable, hence Theorem 2.19 provides us with the decidability result in the following theorem. Undecidability of universality was established in [8]; we give an account of the proof below.

Theorem 2.33. For a timed automaton \( A = (L, \ell_0, F, C, \Sigma, I, E) \), deciding whether \( L(A) = \emptyset \) is PSPACE-complete. It is undecidable whether \( L(A) = T\Sigma^* \).

Proof: We may show that the universality problem for a timed automata is undecidable by reduction from the \( \Sigma \)-hard problem of deciding whether a given 2-counter machine \( M \) has a recurring computation.

Let the timed language \( L_u \) be the set of timed traces encoding recurring computations of \( M \). Observe that \( L_u = \emptyset \) if and only if \( M \) does not have such a computation. We then construct a timed automaton \( A_u \) which accepts the complement of \( L_u \), i.e. \( L(A_u) = T\Sigma^* \setminus L_u \). Hence the language of \( A_u \) is universal if and only if \( M \) does not have a recurring computation.

Recall that a 2-counter, or Minsky, machine \( M \) is a finite sequence of labeled instructions \( \{I_0, \cdots, I_n\} \) and counters \( x_1 \) and \( x_2 \), with \( I_i \), for \( 0 \leq i \leq n - 1 \) on the form
Figure 5. Timed trace encoding a increment instruction $I_{i+1}$ of a 2-counter machine.

$$I_i : x_c := x_c + 1; \text{ goto } I_j \text{ or } I_i : \begin{cases} \text{if } x_c = 0 \text{ then goto } I_j \\ \text{else } x_c = x_c - 1; \text{ goto } I_k \end{cases}$$

for $c \in 1, 2$, with a special $I_n : \text{Halt}$ instruction which stops the computation.

The language $L_u$ is designed such that each $I_i$ and the counters $x_1$ and $x_2$ are represented by actions in $\Sigma$. A correctly encoded computation is represented by a timed trace where “instruction actions” occur at discrete intervals, while the state (values of $x_1$ and $x_2$) is encoded by occurrences of “counter actions” in-between instruction actions (e.g. if $x_i = 5$ after instruction $I_j$, then action $x_i$ occurs 5 times within the succeeding interval of length 1).

When counters are incremented (or decremented), one more (or less) such action occurs through the next interval, and increments and decrements are always from the right. Additionally we require corresponding counter actions to occur exactly with a time difference of 1, such that if $x_i$ occurs with time stamp $a$ then also $x_i$ occurs with time stamp $a + 1$, unless $x_i$ is the rightmost $x_i$ action and $I_i$ at time stamp $[a]$ is a decrement of $x_i$. Figure 5 shows a increment of $x_1$ (from 4 to 5) using actions 1 and 2.

We obtain $A_u$ as a disjunction of timed automata $A^1, \ldots, A^k$ where each $A^i$ violates some property of a (correctly encoded) timed trace in $L_u$, either by accepting traces of incorrect format or inaccurate encodings of instructions.

Consider the instruction: $(p): x_1 := x_1 + 1 \text{ goto } (q)$, incrementing $x_1$ and jumping to $q$. A correct encoding would be similar to the one depicted in Figure 5 where all 1’s and 2’s are matched one time unit later, but with an additional 1 action occurring. In order to accept all traces except this encoding we must consider all possible violations, i.e.

- not incrementing the counter (no change),
- decrementing the counter,
- incrementing the counter more than once,
- jumping to the wrong instruction, or
- incrementing the wrong counter,

and construct a timed automaton having exactly such traces.

Figure 6 shows the timed automaton accepting traces in which instruction $p$ yields no change of $x_1$.

Turning our attention to timed trace inclusion and equivalence, we note the following.
Figure 6. Timed automaton which violates the encoding of the increment instruction.

**Proposition 2.34.** Let $A$ and $B$ be timed automata. If $A$ is timed simulated by $B$, then $L(A) \subseteq L(B)$. If $A$ and $B$ are timed bisimilar, then $L(A) = L(B)$.

By a standard argument, Theorem 2.33 implies undecidability of timed trace inclusion and equivalence, a result first shown in [7].

**Theorem 2.35.** Timed trace inclusion and equivalence are undecidable for timed automata.

There is also a notion of *un timed* traces for timed automata.

**Definition 2.36.** The *un timing* of a set of timed traces $L \subseteq T \Sigma^*$ over a finite set of actions $\Sigma$ is the set

$$UL = \{ w = (a_1, \ldots, a_k) \in \Sigma^* \mid \exists t_1, \ldots, t_k \in \mathbb{R}_{\geq 0} : ((t_1, a_1), \ldots, (t_k, a_k)) \in L \}.$$  

Hence we have a notion of the set $UL(A)$ of un timed language of a timed automaton $A$. One can also define an untime operation $U$ for timed automata, forgetting about the timing information of a timed automaton and thus converting it to a finite automaton; note however that $UL(A) \subset L(UA)$ in general.

**Lemma 2.37 ([8]).** For a timed automaton $A$, $UL(A) = L([A]_{\Sigma})$ provided that $\delta$-transitions in $[A]_{\Sigma}$ are taken as silent.

As a corollary, sets of untimed traces accepted by timed automata are regular:

**Theorem 2.38 ([8]).** For a timed automaton $A = (L, \ell_0, F, C, \Sigma, I, E)$, the set $UL(A) \subseteq \Sigma^*$ is regular. Accordingly, whether $UL(A) = \emptyset$ is decidable, and so is whether $UL(A) = \Sigma^*$. Also untimed trace inclusion and equivalence are decidable.

### 2.6. Zones and difference-bound matrices

As shown in the above sections, regions provide a finite and elegant abstraction of the infinite state space of timed automata, enabling us to prove decidability of reachability, timed and untimed bisimilarity, untimed language equivalence and language emptiness.

Unfortunately, the number of states obtained from the region partitioning is extremely large. In particular, by Proposition 2.18 the number of regions is exponential in the number of clocks as well as in the maximal constants of the timed automaton. Efforts have been made in developing more efficient representations
of the state space $[20, 25, 58]$, using the notion of zones from Definition 2.5 on page 4 as a coarser and more compact representation of the state space.

An extended clock constraint over a finite set $C$ may be represented using a directed weighted graph, where the nodes correspond to the elements of $C$ together with an extra “zero” node $x_0$, and an edge $x_i \rightarrow x_j$ corresponds to a constraint $x_i - x_j \leq k$ (if there is more than one upper bound on $x_i - x_j$, $k$ is the minimum of all these constraints’ right-hand sides). The extra clock $x_0$ is fixed at value 0, so that a constraint $x_i \leq k$ can be represented as $x_i - x_0 \leq k$. Lower bounds on $x_i - x_j$ are represented as (possibly negative) upper bounds on $x_j - x_i$, and strict bounds $x_i - x_j < k$ are represented by adding a flag to the corresponding edge.

The weighted graph in turn may be represented by its adjacency matrix, which in this context is known as a difference-bound matrix or DBM. The above technique has been introduced in [39].

Example 2.3. Figure 7 gives an illustration of an extended clock constraint together with its representation as a difference-bound matrix. Note that the clock constraint contains superfluous information.

Zone-based reachability analysis of a timed automaton $A$ uses symbolic states of the type $(\ell, Z)$, where $\ell$ is a location of $A$ and $Z$ is a zone, instead of the region-based symbolic states of Proposition 2.17.

Definition 2.39. For a finite set $C$, $Z \subseteq \mathbb{R}_{\geq 0}^C$, and $r \subseteq C$, define

- the delay of $Z$ by $Z^\dagger = \{v + d \mid v \in Z, d \in \mathbb{R}_{\geq 0}\}$ and
- the reset of $Z$ under $r$ by $Z[r] = \{v[r] \mid v \in Z\}$.

Lemma 2.40 ([48,72]). If $Z$ is a zone over $C$ and $r \subseteq C$, then $Z^\dagger$ and $Z[r]$ are also zones over $C$.

Extended clock constraints representing $Z^\dagger$ and $Z[r]$ may be computed efficiently (in time cubic in the number of clocks in $C$) by representing the zone $Z$ in a canonical form obtained by computing the shortest-path closure of the directed graph representation of $Z$, see [56].
Figure 8. Canonical representations. Left: shortest-path closure; right: shortest-path reduction.

Example 2.3 (continued). Figure 8 shows two canonical representations of the difference-bound matrix for the zone $Z$ of Figure 7. The left part illustrates the shortest-path closure of $Z$; on the right is the shortest-path reduction [56] of $Z$, essentially obtained by removing redundant edges from the shortest-path closure. The latter is useful for checking zone inclusion, see below.

The zone automaton associated with a timed automaton is similar to the region automaton of Proposition 2.17, but uses zones for symbolic states instead of regions:

**Definition 2.41.** The zone automaton associated with a timed automaton $A = (L, \ell_0, F, C, \Sigma, I, E)$ is the transition system $\llbracket A \rrbracket_z = (S, s_0, \Sigma \cup \{\delta\}, T)$ given as follows:

$S = \{(\ell, Z) \mid \ell \in L, Z \subseteq \mathbb{R}_{\geq 0}^C \text{ zone}\}$

$s_0 = (\ell_0, \llbracket v_0 \rrbracket)$

$T = \{(\ell, Z) \xrightarrow{\delta} (\ell, Z \uparrow \land I(\ell))\}$

$\cup \{(\ell, Z) \xrightarrow{a} (\ell', (Z \land \varphi)[r] \land I(\ell')) \mid \ell \overset{\varphi,a,r}{\longrightarrow} \ell' \in E\}$

The analogue of Proposition 2.13 for zone automata is as follows:

**Proposition 2.42 ([72]).** A state $(\ell, v)$ in a timed automaton $A = (L, \ell_0, F, C, \Sigma, I, E)$ is reachable if and only if there is a zone $Z \subseteq \mathbb{R}_{\geq 0}^C$ for which $v \in Z$ and such that $(\ell, Z)$ is reachable in $\llbracket A \rrbracket_z$.

The zone automaton associated with a given timed automaton is infinite and hence unsuitable for reachability analysis. Finiteness can be enforced by employing normalization, using the fact that region equivalence $\cong$ has finitely many equivalence classes:

**Definition 2.43.** For a timed automaton $A$ and a zone $Z \subseteq \mathbb{R}_{\geq 0}^C$, the normalization of $Z$ is the set \{ $v : C \rightarrow \mathbb{R}_{\geq 0} \mid \exists v' \in Z : v \cong v'$ \}

The normalized zone automaton is defined in analogy to the zone automaton from above, and Proposition 2.42 also holds for the normalized zone automaton.
Hence we can obtain a reachability algorithm by applying any search strategy (depth-first, breadth-first, or another) on the normalized zone automaton.

**Remark 2.44.** For timed automata on extended clock constraints, i.e. with diagonal constraints permitted, it can be shown [24, 27] that normalization as defined above does not give rise to a sound and complete characterization of reachability. Instead, one can apply a refined normalization which depends on the difference constraints used in the timed automaton, see [24].

In addition to the efficient computation of symbolic successor states according to the $\rightarrow$ relation, termination of reachability analysis requires that we can efficiently recognize whether the search algorithm has encountered a given symbolic state. Here it is crucial that there is an efficient way of deciding inclusion $Z_1 \subseteq Z_2$ between zones. Both the shortest-path-closure canonical form as well as the more space-economical shortest-path-reduced canonical form [56], cf. Example 2.3, allow for efficient inclusion checking.

In analogy to difference-bound matrices and overcoming some of their problems, the data structure called clock difference diagram has been proposed [58]. However, the design of efficient algorithms for delay and reset operations over that data structure is a challenging open problem; generally, the design of efficient data structures for computations with (unions of) zones is a field of active research, see [3, 11, 66] for some examples.

### 3. Weighted timed automata

The notion of weighted — or priced — timed automata was introduced independently, at the very same conference, by Behrmann et.al. [18] and Alur et.al. [9]. In these models both edges and locations can be decorated with weights, or prices, giving the cost of taking an action transition or the cost per time unit of delaying in a given location. The total cost of a trace is then simply the accumulated (or total) weight of its discrete and delay transitions.

As a first result, the above two papers independently, and with quite different methods, showed that the problem of cost-optimal reachability is computable for weighted timed automata with non-negative weights. Later, optimal reachability for timed automata with several weight functions was considered in [61] as well as optimal infinite runs in [29, 42].

**Definition 3.1.** A weighted timed automaton is a tuple $A = (L, l_0, F, C, \Sigma, I, E, R, P)$, where $(L, l_0, F, C, \Sigma, I, E)$ is a timed automaton, $R : L \rightarrow \mathbb{Z}$ a location weight-rate mapping, and $P : E \rightarrow \mathbb{Z}$ an edge weight mapping.

The semantics of $A$ is the weighted transition system $[A] = (S, s_0, \Sigma \cup \mathbb{R}_{\geq 0}, T, w)$, where $(S, s_0, \Sigma \cup \mathbb{R}_{\geq 0}, T)$ is the semantics of the underlying timed automaton $(L, l_0, F, C, \Sigma, I, E)$, and the transition weights $w : T \rightarrow \mathbb{R}$ are given as follows:

$$w\left((\ell, v) \xrightarrow{d} (\ell, v + d)\right) = dR(\ell)$$

$$w\left((\ell, v) \xrightarrow{\varphi} (\ell', v')\right) = P(\ell \xrightarrow{\varphi, r} \ell') \text{ with } v \models \varphi, v' = v[r]$$
We shall denote weighted edges and transitions by symbols $\frac{e}{w}$ to illustrate an edge or a transition labeled $e$ with weight $w$.

### 3.1. Optimal reachability

The objective of optimal reachability analysis is to find runs to a final location with the lowest total weight as defined below.

**Example 3.1.** Figure 9 shows a simple weighted timed automaton with final location $\ell_3$. Below we give a few examples of accepting runs, where we identify valuations $v : \{x, y\} \rightarrow \mathbb{R}_{\geq 0}$ with their values $(v(x), v(y))$. The total weights of the runs given here are 17 and 11; actually the second run is optimal in the sense of Problem 3.1 below:

\[
\begin{align*}
(\ell_1, 0, 0) & \xrightarrow{a} (\ell_2, 0, 3) & \xrightarrow{c/4} (\ell_3, 0, 3) \\
(\ell_1, 0, 0) & \xrightarrow{a} (\ell_2, 0, 0) & \xrightarrow{b/6} (\ell_2, 0, 3) & \xrightarrow{c/4} (\ell_3, 0, 3)
\end{align*}
\]

**Definition 3.2.** The total weight of a finite run $\rho = s_0 \xrightarrow{w_1} s_1 \xrightarrow{w_2} \cdots \xrightarrow{w_k} s_k$ in a weighted transition system is $w(\rho) = \sum_{i=1}^{k} w_k$.

We are now in a position to state the problem with which we are concerned here: We want to find accepting runs with minimum total weight in a weighted timed automaton $A$. However due to the possible use of strict clock constraints on edges and in locations of $A$, the minimum total weight might not be realizable, i.e., there might be no run which achieves it. For this reason, one also needs to consider (infinite) sets of runs and the infimum of their members’ total weights:

**Problem 3.1 (Optimal reachability).** Given a weighted timed automaton $A$, compute $W = \inf \{ w(\rho) \mid \rho$ accepting run in $A \}$ and a set $P$ of accepting runs for which $\inf_{\rho \in P} w(\rho) = W$.

The key ingredient in the proof of the following theorem is the introduction of weighted regions in [18]. A weighted region is a region as of Definition 2.16 enriched with an affine cost function describing in a finite manner the cost of reaching any point within it. This notion allows one to define the weighted re-
gion automaton associated with a weighted timed automaton, and one can then show that optimal reachability can be computed in the weighted region automaton. \( \text{PSPACE} \)-hardness in the below theorem follows from \( \text{PSPACE} \)-hardness of reachability for timed automata.

**Theorem 3.3** ([18]). *The optimal reachability problem for weighted timed automata with non-negative weights is \( \text{PSPACE} \)-complete.*

Similar to the notion of regions for timed automata, the number of weighted regions is exponential in the number of clocks as well as in the maximal constants of the timed automaton. Hence a notion of *weighted zone* — a zone extended with an affine cost function — was introduced [55] together with an efficient, symbolic \( A^* \)-algorithm for searching for cost-optimal tracing using branch-and-bound techniques. In particular, efficient means of generalizing the notion of symbolic successor to incorporate the affine cost functions were given.

During the symbolic exploration, several small linear-programming problems in terms of determining the minimal value of the cost function over the given zone have to be dealt with. Given that the constraints of these problems are simple difference constraints, it turns out that substantial gain in performance may be achieved by solving the dual problem of minimum-cost flow [69]. The newly emerged branch UPPAAL CORA provides an efficient tool for cost-optimal reachability analysis, applying the above data structures and algorithms and allowing the user to guide and heuristically prune the search.

### 3.2. Multi-weighted timed automata

The below formalism of doubly weighted timed automata is a generalization of weighted timed automata useful for modeling systems with several different resources.

**Definition 3.4.** A *doubly weighted timed automaton* is a tuple

\[
A = (L, \ell_0, F, C, \Sigma, I, E, R, P)
\]

where \((L, \ell_0, F, C; \Sigma, I, E)\) is a timed automaton, \(R: L \rightarrow \mathbb{Z}^2\) a location weight-rate mapping, and \(P: E \rightarrow \mathbb{Z}^2\) an edge weight mapping.

The semantics of a doubly weighted timed automaton is a doubly weighted transition system defined similarly to Definition 3.1, and the total weight of finite runs is defined accordingly as a pair; we shall refer to the total weights as \(w_1\) and \(w_2\) respectively. These definitions have natural generalizations to *multi-weighted* timed automata with more than two weight coordinates.

The objective of conditional reachability analysis is to find runs to a final location with the lowest total weight in the first weight coordinate while satisfying a constraint on the other weight coordinate.

**Example 3.2.** Figure 10 depicts a simple doubly weighted timed automaton with final location \(\ell_3\). Under the constraint \(w_2 \leq 3\), the optimal run of the automaton can be seen to be
The precise formulation of the conditional optimal reachability problem is as follows, where we again need to refer to (possibly infinite) sets of runs:

**Problem 3.2 (Conditional optimal reachability).** Given a doubly weighted timed automaton $A$ and $M \in \mathbb{Z}$, compute $W = \inf \{ w_1(\rho) \mid \rho \text{ accepting run in } A, w_2(\rho) \leq M \}$ and a set $P$ of accepting runs such that $w_2(\rho) \leq M$ for all $\rho \in P$ and $\inf_{\rho \in P} w(\rho) = W$.

**Theorem 3.5 ([60,61]).** The conditional optimal reachability problem is computable for doubly weighted timed automata with non-negative weights and without weights on edges.

The proof of the above theorem rests on a direct generalization of weighted to doubly-weighted zones. An extension can be found in [61], where it is shown that also the Pareto frontier, i.e. the set of cost vectors which cannot be improved in any cost variable, can be computed.

### 3.3. Optimal infinite runs

In this section we shall be concerned with computing optimal infinite runs in (doubly) weighted timed automata. We shall treat both the limit ratio viewpoint discussed in [29] and the discounting approach of [42].

**Example 3.3.** Figure 11 shows a simple production system modelled as a weighted timed automaton. The system has three modes of production, High, Medium, and Low. The weights model the cost of production, so that the High production mode has a low cost, which is preferable to the high cost of the Low production mode. After operating in a High or Medium production mode for three time units, production automatically degrades (action $d$) to a lower mode. When in Medium or Low production mode, the system can be attended to (action $a$), which advances it to a higher mode.

The objective of optimal-ratio analysis is to find an infinite run in a doubly weighted timed automaton which minimizes the ratio between the two total weights. This will be formalized below.
Figure 11. A weighted timed automaton modelling a simple production system.

**Definition 3.6.** The total ratio of a finite run \( \rho = s_0 \xrightarrow{w_1/z_1} s_1 \xrightarrow{w_2/z_2} \cdots \xrightarrow{w_k/z_k} s_k \) in a doubly weighted transition system is

\[
\Gamma(\rho) = \frac{\sum_{i=1}^{k} w_k}{\sum_{i=1}^{k} z_k}.
\]

The total ratio of an infinite run \( \rho = s_0 \xrightarrow{w_2/z_2} s_1 \xrightarrow{w_3/z_3} \cdots \) is

\[
\Gamma(\rho) = \lim_{k \to \infty} \inf \Gamma(s_0 \rightarrow \cdots \rightarrow s_k).
\]

A special case of optimal-ratio analysis is given by weight-per-time models, where the interest is in minimizing total weight per accumulated time. The example provided in this section is a case of this. In the setting of optimal-ratio analysis, these can be modelled as doubly weighted timed automata with \( R_2(\ell) = 1 \) and \( P_2(e) = 0 \) for all locations \( \ell \) and edges \( e \).

**Example 3.3 (continued).** In the timed automaton of Figure 11, the following cyclic behaviour provides an infinite run \( \rho \):

\[
(H, 0, 0) \xrightarrow{d} (H, 3, 3) \xrightarrow{d} (M, 0, 3) \xrightarrow{d} (M, 3, 6) \xrightarrow{a} (L, 3, 6) \xrightarrow{a} (L, 4, 7) \xrightarrow{a} (M, 0, 0) \xrightarrow{a} (M, 3, 3) \xrightarrow{a} (H, 0, 0) \xrightarrow{a} \cdots
\]

Taking the weight-per-time viewpoint, the total ratio of \( \rho \) is \( \Gamma(\rho) = 4.8 \).

**Problem 3.3 (Minimum infinite ratio).** Given a doubly weighted timed automaton \( A \), compute \( W = \inf \{ \Gamma(\rho) \mid \rho \text{ infinite run in } A \} \) and a set \( P \) of infinite runs for which \( \inf_{\rho \in P} \Gamma(\rho) = W \).

The main tool in the proof of the following theorem is the introduction of the corner-point abstraction of a timed automaton in [29]. This is a finite refinement of the region automaton of Definition 2.16 in which one also keeps track of the corner points of regions. One can then show that any infinite run with minimum ratio must pass through corner points of regions, hence these can be found in the corner-point abstraction by an algorithm first proposed in [53].

The technical condition in the theorem that the second weight coordinate be strongly diverging means that any infinite run \( \rho \) in the closure of the timed automaton in question satisfies \( w_2(\rho) = \infty \), see [29] for details.
Theorem 3.7 ([29]). The minimum infinite ratio problem is computable for doubly weighted timed automata with non-negative and strongly diverging second weight coordinate.

For discount-optimal analysis, the objective is to find an infinite run in a weighted timed automaton which minimizes the discounted total weight as defined below. The point of discounting is that the weight of actions is discounted with time, so that the impact of an event decreases, the further in the future it takes place.

In the definition below, $\varepsilon$ is the empty run, and $(\ell, v) \rightarrow \rho$ denotes the concatenation of the transition $(\ell, v) \rightarrow$ with the run $\rho$.

Definition 3.8. The discounted total weight of finite runs in a weighted timed automaton under discounting factor $\lambda \in [0, 1]$ is given inductively as follows:

$$w_\lambda(\varepsilon) = 0$$

$$w_\lambda((\ell, v) \xrightarrow{a} P \rho) = P + w_\lambda(\rho)$$

$$w_\lambda((\ell, v) \xrightarrow{d} \rho) = R(\ell) \int_0^d \lambda^\tau d\tau + \lambda^d w_\lambda(\rho)$$

The discounted total weight of an infinite run $\rho = (\ell_0, v_0) \xrightarrow{d_1} (\ell_0, v_0 + d_1) \xrightarrow{a_1} P_1 \xrightarrow{\ell_1, v_1} \cdots$ is

$$w_\lambda(\rho) = \lim_{k \to \infty} w_\lambda((\ell_0, v_0) \xrightarrow{\cdots \xrightarrow{a_k} P_k} (\ell_k, v_k))$$

provided that the limit exists.

Example 3.3 (continued). The discounted total weight of the infinite run $\rho$ in the timed automaton of Figure 11 satisfies the following equality, where $I_t = \int_0^t \lambda^\tau d\tau = -\frac{1}{\ln(1 - \lambda)}$:

$$w_\lambda(\rho) = 2I_3 + \lambda^3(5I_3 + \lambda^3(9I_1 + \lambda(1 + 5I_3 + \lambda^3(2 + w_\lambda(\rho))))))$$

With a discount factor of $\lambda = .9$ for example, the discounted total weight of $\rho$ would hence be $w_\lambda(\rho) \approx 40.5$.

Problem 3.4 (Minimum discounted weight). Given a weighted timed automaton $A$ and $\lambda \in [0, 1]$, compute $W = \inf \left\{ w_\lambda(\rho) \mid \rho \text{ infinite run in } A \right\}$ and a set $P$ of infinite runs for which $\inf_{\rho \in P} w_\lambda(\rho) = W$.

The proof of the following theorem rests again on the corner-point abstraction, and on a result in [10]. The technical condition that the timed automaton be time-divergent is analogous to the condition on the second weight coordinate in Theorem 3.7.

Theorem 3.9 ([42]). The minimum discounted weight problem is computable for time-divergent weighted timed automata with non-negative weights and rational $\lambda$. 
4. Timed games

Recently, substantial effort has been made towards the synthesis of winning strategies for timed games with respect to safety and reachability control objectives. From known region-based decidability results, efficient on-the-fly algorithms have been developed [34, 71] and implemented in the newest branch UppAAL Tiga.

For timed games, as for untimed ones, transitions are either controllable or uncontrollable (i.e. under the control of an environment), and the problem is to synthesize a strategy for when to take which (enabled) controllable transitions in order that a given objective is guaranteed regardless of the behaviour of the environment.

**Definition 4.1.** A **timed game** is a tuple \((L, \ell_0, F, C, \Sigma_c, \Sigma_u, I, E)\) with \(\Sigma_c \cap \Sigma_u = \emptyset\) and for which the tuple \((L, \ell_0, F, C, \Sigma = \Sigma_c \cup \Sigma_u, I, E)\) is a timed automaton.

Edges with actions in \(\Sigma_c\) are said to be **controllable**, those with actions in \(\Sigma_u\) are **uncontrollable**.

**Example 4.1.** Figure 12 provides a simple example of a timed game. Here, \(\Sigma_c = \{c_1, c_2, c_4\}\) and \(\Sigma_u = \{u_1, u_2, u_3\}\), and the controllable edges are drawn with solid lines, the uncontrollable ones with dashed lines.

We need the notion of **strategy**; essentially, a strategy provides instructions for which controllable edge to take, or whether to wait, in a given state:

**Definition 4.2.** A **strategy** for a timed game \(A = (L, \ell_0, F, C, \Sigma_c, \Sigma_u, I, E)\) is a mapping \(\sigma\) from finite runs of \(A\) to \(\Sigma_c \cup \{\delta\}\), where \(\delta \notin \Sigma\), such that for any run \(\rho = (\ell_0, v_0) \rightarrow \cdots \rightarrow (\ell_k, v_k)\),

- if \(\sigma(\rho) = \delta\), then \((\ell, v) \xrightarrow{d} (\ell, v + d)\) in \([A]\) for some \(d > 0\), and
- if \(\sigma(\rho) = a\), then \((\ell, v) \xrightarrow{a} (\ell', v')\) in \([A]\).

A strategy \(\sigma\) is said to be **memoryless** if \(\sigma(\rho)\) only depends on the last state of \(\rho\), i.e. if \(\rho_1 = (\ell_0, v_0) \xrightarrow{d_1} (\ell_0, v_0 + d_1) \rightarrow \cdots \rightarrow (\ell_k, v_k)\), \(\rho_2 = (\ell_0, v_0) \xrightarrow{d_2} (\ell_0, v_0 + d'_1) \rightarrow \cdots \rightarrow (\ell_k, v_k)\) imply \(\sigma(\rho_1) = \sigma(\rho_2)\).
An outcome of a strategy is any run which adheres to its instructions in the obvious manner:

**Definition 4.3.** A run \((\ell_0, v_0) \xrightarrow{d_1} (\ell_0, v_0 + d_1) \rightarrow \cdots \rightarrow (\ell_k, v_k)\) in a timed game \(A = (L, \ell_0, F, C, \Sigma_c, \Sigma_u, I, E)\) is said to be an outcome of a strategy \(\sigma\) provided that

- for all \((\ell_i, v_i) \xrightarrow{d} (\ell_i, v_i + d)\) and for all \(d' < d\), we have \(\sigma((\ell_0, v_0) \rightarrow \cdots \rightarrow (\ell_i, v_i + d')) = \delta\), and
- for all \((\ell_i, v_i + d) \xrightarrow{a} (\ell_{i+1}, v_{i+1})\) for which \(a \in \Sigma_c\), we have \(\sigma((\ell_0, v_0) \rightarrow \cdots \rightarrow (\ell_i, v'_i)) = a\).

An outcome is said to be maximal if \(\ell_k \in F\), or if \((\ell_k, v_k) \xrightarrow{a} (\ell_{k+1}, v_{k+1})\) implies \(a \in \Sigma_u\).

Hence an outcome is maximal if it stops in a final state, or if no controllable actions are available at its end. An underlying assumption is that uncontrollable actions cannot be forced, hence a maximal outcome which does not end in a final state may "get stuck" in a non-final state. The aim of reachability games is to find strategies all of whose maximal outcomes end in a final state; the aim of safety games is to find strategies all of whose (not necessarily maximal) outcomes avoid final states:

**Definition 4.4.** A strategy is said to be winning for the reachability game if any of its maximal outcomes is an accepting run. It is said to be winning for the safety game if none of its outcomes are accepting.

**Example 4.1 (continued).** The following memoryless strategy is winning for the reachability game on the timed game from Figure 12:

\[
\begin{align*}
\sigma(\ell_1, v) &= \begin{cases} 
\delta & \text{if } v(x) \neq 1 \\
\ c_1 & \text{if } v(x) = 1 
\end{cases} \\
\sigma(\ell_2, v) &= \begin{cases} 
\delta & \text{if } v(x) < 2 \\
\ c_2 & \text{if } v(x) \geq 2 
\end{cases} \\
\sigma(\ell_3, v) &= \begin{cases} 
\delta & \text{if } v(x) < 1 \\
\ c_3 & \text{if } v(x) \geq 1 
\end{cases} \\
\sigma(\ell_4, v) &= \begin{cases} 
\delta & \text{if } v(x) \neq 1 \\
\ c_4 & \text{if } x(x) = 1 
\end{cases}
\end{align*}
\]

**Problem 4.1 (Reachability and safety games).** Given a timed game \(A\), does there exist a winning strategy for the reachability game on \(A\)? Does there exist a winning strategy for the safety game on \(A\)?

An important ingredient in the proof of the following theorem is the fact that for reachability as well as safety games, it is sufficient to consider memoryless strategies. This is not the case for other, more subtle, control objectives (e.g. counting properties modulo some \(N\)) as well as for the synthesis of winning strategies under partial observability.

**Theorem 4.5 ([12, 65]).** The reachability and safety games are decidable for timed games.
In [35] the on-the-fly algorithm applied in UPPAAL TIGA has been extended to timed games under partial observability.

The field of timed games is a very active research area. Research has been conducted towards the synthesis of optimal winning strategies for reachability games on weighted timed games. In [5,30] computability of optimal strategies is shown under a certain condition of strong cost non-zenoness, requiring that the total weight diverges with a given minimum rate per time. Later undecidability results [28,33] show that for weighted timed games with three or more clocks this condition (or a similar one) is necessary. Lately [31] proves that optimal winning strategies for one-clock weighted timed games, though there is an unsettled (large) gap between the known lower bound complexity $\text{P}$ and an upper bound of $\text{3EXPTIME}$.

We conclude this section by reestablishing the connection between the notion of games and bisimulation [70] in the presence of time:

**Proposition 4.6.** Timed bisimilarity polynomial-time reduces to timed safety games.

Observe that this provides an alternative proof of the decidability of timed bisimilarity in Theorem 2.23 on page 9.

**Proof:** Given timed automata $A_i = (L_i, \ell_0, F, C_i, \Sigma, I_i, E_i)$, for $i \in \{1,2\}$, with $C_1 \cap C_2 = \emptyset$, we consider the timed game with locations $L = \{\bot\} \cup (L_1 \times L_2) \cup (L_1 \times L_2 \times \Sigma \times \{1,2\})$, where $F = \{\bot\}$ is a designated final location. We set $C = C_1 \cup C_2 \cup \{z\}$, where $z \notin C_1 \cup C_2$ is a fresh clock, $\Sigma_c = \Sigma \cup \{\bot\}$ and $\Sigma_u = \{a' \mid a \in \Sigma\}$, and $E$ is defined by

$$
(p,q) \xrightarrow{\varphi,a,r} (p',q,a) \in E \quad \text{if} \quad p \xrightarrow{\varphi,a,r} p' \in E_1,
$$

$$
(p,q) \xrightarrow{\varphi,a,r} (p',q,a) \in E \quad \text{if} \quad q \xrightarrow{\varphi,a,r} q' \in E_2,
$$

$$
(p',q,a) \xrightarrow{\varphi,a,r} (p',q') \in E \quad \text{if} \quad q \xrightarrow{\varphi,a,r} q' \in E_2,
$$

$$
(p,q,a) \xrightarrow{\varphi,a,r} (p',q') \in E \quad \text{if} \quad p \xrightarrow{\varphi,a,r} p' \in E_1,
$$

$$
(p,q,a) \xrightarrow{\Phi \land z=0,1} \bot \quad \text{for all} \quad i \in \{1,2\}.
$$

Here we denote $\tilde{r} = r \cup \{z\}$ and $\tilde{\varphi} = \varphi \land z = 0$, and $\Phi = \bigvee_j \neg \varphi_j$ when $q \xrightarrow{\varphi_j,a,r} q'$ and $i = 1$ and symmetrically for $i = 2$. Location invariants are defined by $I(p,q) = I_1(p) \land I_2(q)$ for $(p,q) \in L_1 \times L_2$ and $I(p,q,a) = (z = 0)$ for all $(p,q,a) \in L_1 \times L_2 \times \Sigma \times \{1,2\}$. See Figure 13 for a simple example of this construction.

It remains to be seen that $A_1$ and $A_2$ are timed bisimilar if and only if a strategy $\sigma$ exists for which any outcome $\rho = (\ell_0, v_0) \xrightarrow{d_1} (\ell_0, v_0 + d_1) \cdots \xrightarrow{d_k} (\ell_k, v_k)$ satisfies $\ell_k \neq \bot$ (i.e. it avoids the final location $\bot$).

Assume $A_1$ and $A_2$ are timed bisimilar, then we can prove something stronger than the above, namely that any strategy will avoid $\bot$. Indeed, if $(\ell_0, v_0) \xrightarrow{d_1}$
Figure 13. A timed game constructed for bisimilarity checking of two simple timed automata

\[(\ell_0, v_0 + d_1) \rightarrow \cdots \rightarrow (\ell_k, v_k)\] is a run in the timed game, and the transition \[(\ell_{k-2}, v_{k-2}) \xrightarrow{a} (\ell_{k-1}, v_{k-1})\] exists due to the first component \((p_{k-2}, q_{k-2})\) of \((\ell_{k-2}, v_{k-2})\), then the corresponding \(\xrightarrow{a}\) transition in \(A_1\) has a matching \(\xrightarrow{a}\) transition in \(A_2\). Hence the state \((\ell_{k-1}, v_{k-1})\) has an enabled \(a\)-labeled edge, which by definition of \(\Phi\) implies that the edge to \(\bot\) is disabled, thus \(\ell_k \neq \bot\). A symmetric argument applies in the other case.

Now assume \(\sigma\) is a strategy which ensures avoidance of \(\bot\), then we shall show that any \((\ell_j, v) = ((p_j, q_j), v)\) for \(j \geq 0\), \((i.e.\) of type \(S_1 \times S_2\)) occurring in an outcome of \(\sigma\) satisfies \((p_j, v) \sim (q_j, v)\). Assume to the contrary that \((p_j, v) \xrightarrow{a} (p'_j, v') \in [A_1]\) and \((q_j, v) \xrightarrow{b} (q'_j, v'') \in [A_2]\) for some \(a \in \Sigma\), then we may extend the run to \((\ell_j, v)\) by \((\ell_j, v) \xrightarrow{a} (\ell_{j+1}, v') \xrightarrow{\bot} \bot\), moreover \(\sigma(\{(\ell_0, v_0) \rightarrow \cdots \rightarrow (\ell_{j+1}, v')\}) = \bot\) is the only choice for \(\sigma\) as by construction neither \(\delta\) (due to the invariant \(z = 0\)) nor any \(b \neq a\) is available. □

References


