The untiming construction for timed automata

B. Srivathsan
Chennai Mathematical Institute, India

In this document we explain the following theorem, which was proved in [AD94].

Theorem 1 *For every timed automaton $\mathcal{A}$, there exists a finite automaton accepting its untimed language* Untime($\mathcal{L}(\mathcal{A})$).

A naïve approach is to consider the finite automaton obtained by removing the guards and resets from $\mathcal{A}$. This is incorrect: consider a timed automaton $\mathcal{A}_0$ consisting of two transitions: $q_0 \xrightarrow{a,\ x>2} q_1 \xrightarrow{a,\ x<1} q_2$, with q_0 being the initial state and q_2 the final state. The timed language $\mathcal{L}(\mathcal{A}_0)$ is empty, whereas the language of the finite automaton obtained by the naïve construction is $\{aa\}$. This example illustrates that in order to detect the untimed sequences that have a timed extension, we need to track the clock values at each stage of the sequence. In the next sections, we will see how this can be done using finitely many states. We start this document with some preliminary notions and then continue to describing two equivalence relations which lead to the construction of the final finite state automaton.

1 Preliminaries

Let $\mathbb{R}_{\geq 0}$ denote the set of non-negative reals. A clock is a variable that ranges over $\mathbb{R}_{\geq 0}$. Let X be a set of clocks. A *clock constraint* ϕ is a conjunction of comparisons of a clock with a constant, given by the following grammar:

$$\phi := x \sim c \mid \phi \wedge \phi$$

where $x \in X$, $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$. For example, $(x \leq 3 \wedge y > 0)$ is a clock constraint. Let $\Phi(X)$ denote the set of clock constraints over the set of clocks X.

Remark 2 Notice that we have chosen only integer constants in the clock constraints.

Definition 3 (Clock valuation) A *clock valuation* over X is a function $v : X \to \mathbb{R}_{\geq 0}$ that maps each clock to a non-negative real. The set of all clock valuations is denoted by $\mathbb{R}_{\geq 0}^X$.

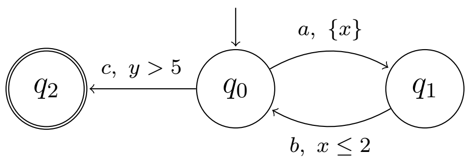


Figure 1.1: A timed automaton $\mathcal{A}_1$

For example, $\langle x = 0.45, y = 7.7, z = 15\rangle$ is a valuation when $X = \{x, y, z\}$. When the set of clocks and the order are clear from the context, we will just write $\langle 0.45, 7.7, 15\rangle$.

We denote by $\mathbf{0}$ the valuation that associates 0 to every clock in X. A valuation v is said to satisfy a constraint ϕ, written as $v \vDash \phi$, when every constraint in ϕ holds after replacing every x by $v(x)$. For example, $\langle x = 0.4, y = 9.3, z = 4.6\rangle \models (x < 5) \wedge y > 4$, but $\langle x = 0.4, y = 9.3, z = 4.6\rangle \not\models (z \leq 4)$.

For $\delta \in \mathbb{R}_{\geq 0}$, let $v + \delta$ be the valuation that associates $v(x) + \delta$ to every clock x. For instance, $\langle 0.4, 9.3, 4.6\rangle + 5.1 = \langle 5.5, 14.4, 9.7\rangle$. We will use this notation to talk about the valuations that arise after a δ time-elapse from v.

For $R \subseteq X$, let $[R]v$ be the valuation that sets x to 0 if $x \in R$, and that sets x to $v(x)$ otherwise. For example, $[\{x, y\}]\langle 0.4, 9.3, 4.6\rangle = \langle 0, 0, 4.6\rangle$. This notation would be used to denote the valuation that is obtained from v after resetting clocks in R.

For a valuation v and a clock x, we denote the integral part of $v(x)$ by $\lfloor v(x)\rfloor$ and the fractional part of $v(x)$ by $\{v(x)\}$. We write $\triangleleft$ to mean either $\leq$ or $<$, and $\triangleright$ to mean either $\geq$ or $>$.

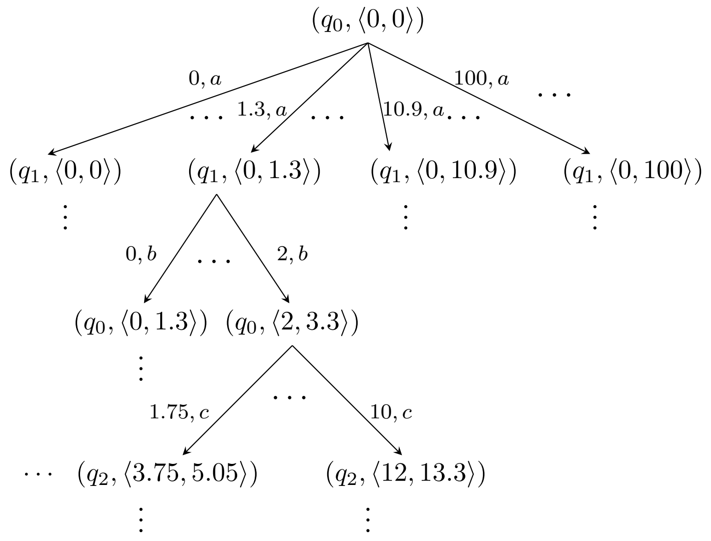
Let us start with a formal definition of a timed automaton.

Definition 4 (Timed automaton [AD94]) A *Timed Automaton (TA)* is a tuple $\mathcal{A} = (Q, \Sigma, X, T, q_0, Acc)$ where Q is a finite set of states, Σ is a finite alphabet, X is a finite set of clocks, $q_0 \in Q$ is the initial state, $Acc \subseteq Q$ is a set of accepting (final) states, $T \subseteq Q \times \Sigma \times \Phi(X) \times 2^X \times Q$ is a finite set of transitions (q, a, g, R, q') where a is a letter in Σ, g is a clock constraint called the *guard*, and R is the set of clocks that are *reset* on the transition. We will call these transitions in a timed automaton as *edges*.

Remark 5 For technical convenience, we choose to have a single initial state $q_0 \in Q$ instead of a set of initial states $Q_0 \subseteq Q$. For the algorithm that we propose, this does not make any difference. One can easily extrapolate it to the case of multiple initial states.

Figure 1.1 gives an example of a time automaton. The behaviour of a timed automaton is described by an infinite transition system[1] as illustrated in Figure 1.2. We will call it the *semantics* of a timed automaton. For example, the transition $(q_1, \langle 0, 1.3\rangle) \xrightarrow{2,b} (q_0, \langle 2, 3.3\rangle)$ means that when the timed automaton is in state q_1 with values of clocks $\langle 0, 1.3\rangle$, it can elapse 2 units of time and fire the edge $(q_1, b, x \leq 2, \{\}, q_0)$ and go to the configuration $(q_0, \langle 2, 3.3\rangle)$. Here is the formal definition of this infinite transition system.

[1]A transition system is essentially an automaton, except that we do not associate any language to it. We only care about the states and transitions between them.



$(q_0, \langle 0,0\rangle)$
$0, a$
$1.3, a$
$10.9, a$
$100, a$
$(q_1, \langle 0,0\rangle)$
$(q_1, \langle 0,1.3\rangle)$
$(q_1, \langle 0,10.9\rangle)$
$(q_1, \langle 0,100\rangle)$
$0, b$
$2, b$
$(q_0, \langle 0,1.3\rangle)$
$(q_0, \langle 2,3.3\rangle)$
$1.75, c$
$10, c$
$(q_2, \langle 3.75,5.05\rangle)$
$(q_2, \langle 12,13.3\rangle)$

Figure 1.2: Part of the transition system showing the behaviours of the timed automaton $\mathcal{A}_1$ of Figure 1.1

Definition 6 (Semantics of a timed automaton) Let $\mathcal{A}$ be a timed automaton. The semantics of $\mathcal{A}$ is given by a transition system $\mathcal{S}_\mathcal{A}$ whose nodes are configurations (q, v) consisting of a state q of $\mathcal{A}$ and a valuation v giving the values of clocks. The initial configuration is given by $(q_0, \mathbf{0})$ with q_0 being the initial state of $\mathcal{A}$. The transitions are labeled with letters from $\mathbb{R}_{\geq 0} \times \Sigma$. There is a transition $(q, v) \xrightarrow{\delta,a} (q_1, v_1)$ if there exists an edge (q, a, g, R, q_1) in the timed automaton such that $v + \delta \models g$ and $v_1 = [R]v$.

Definition 7 (Timed words, timed runs and timed language) A timed word is a pair (w, τ) consisting of a word $w = a_0a_1 \cdots a_k \in \Sigma^*$ and a time sequence $\tau = \tau_0\tau_1 \cdots \tau_k$ of non-negative reals such that $\tau_0 \leq \tau_1 \leq \cdots \leq \tau_k$.

A *run* of $\mathcal{A}$ on a timed word $(a_0a_1 \cdots a_k, \tau_0\tau_1 \cdots \tau_k)$ is a path in $\mathcal{S}_\mathcal{A}$ starting from $(q_0, \mathbf{0})$:

$$(q_0, v_0) \xrightarrow{\delta_0,a_0} (q_1, v_1) \xrightarrow{\delta_1,a_1} (q_2, v_2) \xrightarrow{\delta_2,a_2} \ldots \xrightarrow{\delta_k,a_k} (q_{k+1}, v_{k+1})$$

such that $\delta_0 = \tau_0$ and $\delta_i = \tau_i - \tau_{i-1}$ for $1 \leq i \leq k$. The run is accepting if q_{k+1} is an accepting state.

The *timed language* of the timed automaton, denoted as $\mathcal{L}(\mathcal{A})$, is the set of timed words (w, τ) such that $\mathcal{A}$ has an accepting run on (w, τ).

For example the timed language $\mathcal{L}(\mathcal{A}_1)$ of the timed automaton in Figure 1.1 is the union of timed words (c, τ) with $\tau > 5$ along with timed words of the form $((ab)^kc, \tau_1\tau_1'\tau_2\tau_2' \cdots \tau_k\tau_k'\tau)$ with $k \geq 1$ such that $\tau_i' - \tau_i \leq 2$ for all $1 \leq i \leq k$ and $\tau > 5$.

Definition 8 (Untimed language) For a timed language L over Σ, we define $\mathrm{Untime}(L) := \{w \in \Sigma^* \mid (w, \tau) \in L\}$.

Notice that for a timed automaton $\mathcal{A}$, the language $\mathrm{Untime}(\mathcal{L}(\mathcal{A}))$ is the set of sequences $a_0, a_1, \ldots, a_k$ such that there is a path $(\delta_0, a_0)(\delta_1, a_1) \cdots (\delta_k, a_k)$ in $\mathcal{S}_\mathcal{A}$. Therefore, if we erase the δ part in the transitions of $\mathcal{S}_{\mathcal{A}}$ and look at the underlying system, it will be an infinite automaton for Untime($\mathcal{L}(\mathcal{A})$) where there is an edge $(q,v) \xrightarrow{a} (q_1, v_1)$ if there exists a δ such that $(q,v) \xrightarrow{\delta,a} (q_1,v_1)$ in $\mathcal{S}_{\mathcal{A}}$. In the next sections, we will try to merge valuations into buckets so that the untimed behaviour of $\mathcal{S}_{\mathcal{A}}$ can be mimicked, and yet we have only finitely many buckets.

2 Neighbourhood equivalence

As a first task we divide the space $\mathbb{R}^X_{\geq 0}$ into units that are indistinguishable by guards containing integer constants. These units will be called neighbourhoods.

Definition 9 (Neighbourhood equivalence) Two valuations v and v' are said to be neighbourhood equivalent, written as $v \simeq_{\mathsf{nbd}} v'$ if:

1. $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ for all clocks x,
2. $\{v(x)\} = 0$ iff $\{v'(x)\} = 0$ for all clocks x
3. for every pair of clocks x, y:
   (a) $\{v(x)\} < \{v(y)\} \Leftrightarrow \{v'(x)\} < \{v'(y)\}$
   (b) $\{v(x)\} = \{v(y)\} \Leftrightarrow \{v'(x)\} = \{v'(y)\}$

Each equivalence class of $\simeq_{\mathsf{nbd}}$ will be called a *neighbourhood.* We write $\mathsf{nbd}(v)$ for the neighbourhood of v.

Remark 10 Notice that the condition 3 with two sub-conditions (a) and (b) can be replaced with a single condition: for every pair of clocks x, y, we have $\{v(x)\} \leq \{v(y)\} \Leftrightarrow \{v'(x)\} \leq \{v'(y)\}$. Proving that this replacement results in an equivalent definition is left as an exercise.

Lemma 11 Let $v \simeq_{\mathsf{nbd}} v'$. For every $\delta \geq 0$, there exists a $\delta' \geq 0$ such that $v + \delta \simeq_{\mathsf{nbd}} v' + \delta'$.

Proof
First choose $\lfloor \delta' \rfloor$ to be $\lfloor \delta \rfloor$. This choice ensures $v + \lfloor \delta \rfloor \simeq_{\mathsf{nbd}} v' + \lfloor \delta' \rfloor$. Let $x_1 \lessdot_1 x_2 \lessdot_2 \cdots \lessdot_{k-1} x_k$ be the ordering of fractional parts in both the valuations, where $\lessdot$ denotes either $<$ or $=$. This ordering is the same in both v and v' due to the third condition in the definition of $\simeq_{\mathsf{nbd}}$.

From $v + \lfloor \delta \rfloor$, elapsing a fractional amount $\{\delta\}$ might move some of the clocks up to the next integer. Let $x_j, x_{j+1}, \dots, x_k$ be the clocks that have their integral values increased from $v + \lfloor \delta \rfloor$ due to the fractional elapse $\{\delta\}$. Thanks to the denseness of the real line, one can choose $\{\delta'\}$ between the fractional values of clocks x_{j-1} and x_j in $v' + \lfloor \delta' \rfloor$ so that $v' + \delta'$ has the same integers as $v + \delta$, the same clocks have fractional parts zero, and finally the same ordering of fractional parts between $v + \delta$ and $v' + \delta'$. $\square$

Lemma 12 Let $v \simeq_{\mathsf{nbd}} v'$. Then, $v \models \phi$ iff $v' \models \phi$ for every clock constraint $\phi \in \Phi(X)$.

Proof
This follows due to the first two conditions in the definition of $\simeq_{\mathsf{nbd}}$. $\square$

Lemma 13 Let $v \simeq_{\mathsf{nbd}} v'$. Then, $[R]v \simeq_{\mathsf{nbd}} [R]v'$ for every subset R of clocks.

Proof
Follows by definition, and the hypothesis $v \simeq_{\mathsf{nbd}} v'$. $\square$

The above three lemmas give the following property of the semantics $\mathcal{S}_{\mathcal{A}}$.

Proposition 14 Let $(q, v) \xrightarrow{\delta,a} (q_1, v_1)$ be a transition in $\mathcal{S}_{\mathcal{A}}$. For every $v' \in \mathsf{nbd}(v)$, there exists a transition $(q, v') \xrightarrow{\delta',a} (q_1, v_1')$ in $\mathcal{S}_{\mathcal{A}}$ such that $v_1' \in \mathsf{nbd}(v_1)$.

Definition 15 (Neighbourhood automaton) Given a timed automaton $\mathcal{A}$, we define an infinite state automaton $\mathsf{nbdAutomaton}(\mathcal{A})$ called the neighbourhood automaton. Its states are pairs $(q, \mathsf{nbd}(v))$ for every state q of $\mathcal{A}$ and every valuation $v \in \mathbb{R}^X_{\geq 0}$. For every $(q, v) \xrightarrow{\delta,a} (q_1, v_1)$ in $\mathcal{S}_{\mathcal{A}}$, there is a transition $(q, \mathsf{nbd}(v)) \xrightarrow{a} (q_1, \mathsf{nbd}(v_1))$ in $\mathsf{nbdAutomaton}(\mathcal{A})$. Initial state is $(q_0, \mathsf{nbd}(\mathbf{0}))$ where q_0 is the initial state of $\mathcal{A}$. Final states are $(q, \mathsf{nbd}(v))$ where q is a final state of $\mathcal{A}$.

Proposition 16 For every run $(q_0, \mathsf{nbd}(\mathbf{0})) \xrightarrow{a_0} (q_1, \mathsf{nbd}(v_1)) \xrightarrow{a_1} \cdots \xrightarrow{a_n} (q_{n+1}, \mathsf{nbd}(v_{n+1}))$ of $\mathsf{nbdAutomaton}(\mathcal{A})$ there exists a run $(q_0, \mathbf{0}) \xrightarrow{\delta_0,a_0} (q_1, v_1') \xrightarrow{\delta_1,a_1} (q_2, v_2') \xrightarrow{\delta_2,a_2} \cdots \xrightarrow{\delta_n,a_n} (q_{n+1}, v_{n+1}')$ such that $v_i' \in \mathsf{nbd}(v_i)$ for all $0 \leq i \leq n+1$.

Proof
Proof proceeds by induction on i. For the base case $i = 0$, notice that $(q_0, \mathbf{0})$ with $\mathbf{0} \in \mathsf{nbd}(\mathbf{0})$ exists in $\mathcal{S}_{\mathcal{A}}$.

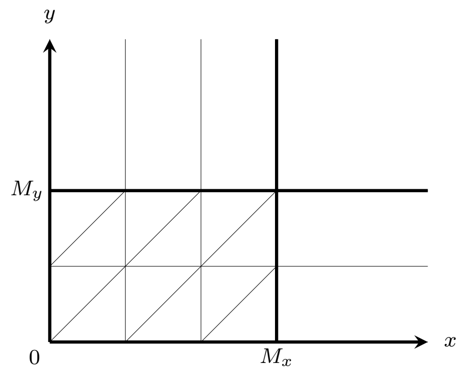
Assume that we have proved the lemma for $i = k$, that is we have constructed a run $(q_0, \mathbf{0}) \xrightarrow{\delta_0,a_0} (q_1, v_1') \xrightarrow{\delta_2,a_2} \cdots \xrightarrow{\delta_{k-1},a_{k-1}} (q_k, v_k')$. Therefore, there is a run in $\mathcal{S}_{\mathcal{A}}$ leading to $(q_k, u_k$ with $u_k \in \mathsf{nbd}(v_k)$. By definition of the transitions in $\mathsf{nbdAutomaton}(\mathcal{A})$, the presence of a transition $(q_k, \mathsf{nbd}(v_k)) \xrightarrow{a_k} (q_{k+1}, \mathsf{nbd}(v_{k+1}))$ entails that there exists a transition $(q_k, \overline{v}_k) \xrightarrow{\delta,a_k} (q_{k+1}, \overline{v}_{k+1})$ in $\mathcal{S}_{\mathcal{A}}$ such that $\overline{v}_k \in \mathsf{nbd}(v_k)$ and $\overline{v}_{k+1} \in \mathsf{nbd}(v_{k+1})$. Then, from Proposition 14, there exists a transition $(q_k, v_k') \xrightarrow{\delta',a_k} (q_{k+1}, v_{k+1}')$ such that $v_{k+1}' \in \mathsf{nbd}(v_{k+1})$. This gives an extension to the run obtained by the induction hypothesis, thereby proving the induction step. $\square$

Proposition 17 For every run $(q_0, \mathbf{0}) \xrightarrow{\delta_0,a_0} (q_1, v_1) \xrightarrow{\delta_1,a_1} \cdots \xrightarrow{\delta_n,a_n} (q_n, v_n)$ of $\mathcal{S}_{\mathcal{A}}$ there exists a run $(q_0, \mathsf{nbd}(\mathbf{0})) \xrightarrow{a_0} (q_1, \mathsf{nbd}(v_1)) \xrightarrow{a_1} \cdots \xrightarrow{a_n} (q_n, \mathsf{nbd}(v_n))$ in $\mathsf{nbdAutomaton}(\mathcal{A})$.

Proof
This follows directly from the definition of transitions in $\mathsf{nbdAutomaton}(\mathcal{A})$. $\square$

Propositions 16 and 17 lead to the following theorem.

Theorem 18 *The infinite state automaton* $\mathsf{nbdAutomaton}(\mathcal{A})$ *accepts* $\mathrm{Untime}(\mathcal{L}(\mathcal{A}))$.



y
M_y
0
M_x
x
12 corner points, e.g. (2,1)
30 open line segments, e.g. $1 < x = y < 2$
18 open regions, e.g. $0 < x < y < 1$

Figure 1.3: Division into regions with two clocks x and y [AD94].

The $\mathsf{nbdAutomaton}(\mathcal{A})$ is in some sense better than $\mathcal{S}_\mathcal{A}$ since its alphabet is Σ and it has countably many states, unlike $\mathcal{S}_\mathcal{A}$ whose alphabet was $\mathbb{R}_{\geq 0} \times \Sigma$ and whose states were uncountably infinite. In the next section, we perform a further merge of neighbourhoods into finitely many equivalence classes called regions.

3 Region equivalence

Let X be a finite set of clocks. Let $M : X \mapsto \mathbb{N} \cup \{-\infty\}$ be a *bound function* that associates a constant $M_x \in \mathbb{N}$ to every clock x. This is a slight generalization from what we saw in class, where considered a single constant M for all clocks.

Definition 19 (Region equivalence [AD94]) Two valuations $v, v' \in \mathbb{R}_{\geq 0}^X$ are *region equivalent* w.r.t. M, denoted $v \sim_M v'$ iff for every $x, y \in X$:

1. $v(x) > M_x$ iff $v'(x) > M_x$;
2. if $v(x) \leq M_x$, then $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$;
3. if $v(x) \leq M_x$, then $\{v(x)\} = 0$ iff $\{v'(x)\} = 0$;
4. if $v(x) \leq M_x$ and $v(y) \leq M_y$ then $\{v(x)\} \leq \{v(y)\}$ iff $\{v'(x)\} \leq \{v'(y)\}$.

Given an automaton $\mathcal{A}$, a bound function is obtained by choosing for a clock x, the maximum constant appearing in a guard involving x. Then, the first three conditions in the above definition ensure that the two valuations satisfy the same guards. The last one enforces that for every $\delta \in \mathbb{R}_{\geq 0}$ there is $\delta' \in \mathbb{R}_{\geq 0}$, such that valuations $v + \delta$ and $v' + \delta'$ satisfy the same guards.

Definition 20 (Region [AD94]) Let $M : X \mapsto \mathbb{N} \cup \{-\infty\}$ be a bound function. A *region* is an equivalence class of $\sim_M$. We write $[v]^M$ for the region of v, and $\mathcal{R}_M$ for the set of all regions with respect to M.

Figure 1.3 shows the division into regions when there are two clocks x and y. We also give below a constructive definition of regions which would be useful to estimate the number of regions.

Definition 21 (Region: constructive definition [AD94]) A region with respect to bound function M is a set of valuations specified as follows:

1. for each clock $x \in X$, one constraint from the set:
   $$\{x = c \mid c = 0, \dots, M_x\} \cup \{c - 1 < x < c \mid c = 1, \dots, M_x\} \cup \{x > M_x\}$$
2. for each pair of clocks x, y having interval constraints: $c-1 < x < c$ and $d-1 < y < d$, it is specified if $\{x\}$ is less than, equal to or greater than $\{y\}$.

If r is a region then we will write $r \vDash g$ to mean that every valuation in r satisfies the guard g. It is straightforward to see that if a valuation $v \in r$ satisfies the guard g, then every valuation $v' \in r$ satisfies g. We will now show the other property with respect to time-elapse mentioned above.

Lemma 22 Let v, v' be valuations such that $v' \sim_M v$. Then, for all $\delta \in \mathbb{R}_{\geq 0}$, there exists a $\delta' \in \mathbb{R}_{\geq 0}$ such that $v' + \delta' \sim_M v + \delta$.

Proof
We know $v' \sim_M v$ and we are given δ. We need to choose δ'. Put $\lfloor \delta' \rfloor$ to be $\lfloor \delta \rfloor$. Clearly, we have $v' + \lfloor \delta' \rfloor \sim_M v + \lfloor \delta \rfloor$: that is, valuations $v' + \lfloor \delta' \rfloor$ and $v + \lfloor \delta \rfloor$ have the same integral parts and the same ordering of fractional parts (modulo M). Let $x_1 \triangleleft_1 x_2 \triangleleft_2 \dots \triangleleft_{k-1} x_k$ be the ordering of fractional parts of clocks less than M in both the valuations. Here $\triangleleft$ denotes either $<$ or $=$.

From $v + \lfloor \delta \rfloor$, elapsing a fractional amount $\{\delta\}$ might move some of the clocks up to the next integer. Let $x_j, x_{j+1}, \dots, x_k$ be the clocks that have their integral values increased from $v + \lfloor \delta \rfloor$ due to the fractional elapse $\{\delta\}$. Thanks to the denseness of the real line, one can choose $\{\delta'\}$ between the fractional values of clocks x_{j-1} and x_j in $v' + \lfloor \delta' \rfloor$ so that $v' + \delta'$ has the same integers as $v + \delta$ and the same ordering of fractional parts (modulo M). □

Given a bound function M, the number of regions in $\mathcal{R}_M$ is finite. Once this finite partition of the valuations is obtained, we proceed to define a finite graph built from these regions, that captures the behaviour of the timed automaton.

For an automaton $\mathcal{A}$, to define its region graph, we consider a bound function $M_{\mathcal{A}}$ that is obtained from the automaton's definition.

Definition 23 (Maximal bounds) Given an automaton $\mathcal{A}$, the *maximal bounds function* $M_{\mathcal{A}} : X \mapsto \mathbb{N} \cup \{-\infty\}$ associates to each clock x the biggest constant appearing in a guard of the automaton that involves x. If there is no guard involving x, then $M_{\mathcal{A}}(x)$ is assigned $-\infty$.

We define the region automaton of $\mathcal{A}$ using the $\sim_{M_{\mathcal{A}}}$ relation.

Definition 24 (Region automaton [AD94]) States of the *region automaton* are of the form (q, r) for q a state of $\mathcal{A}$ and $r \in \mathcal{R}_{M_{\mathcal{A}}}$ a region. There is a transition $(q, r) \xrightarrow{a} (q', r')$ if there is a transition $t := (q, a, g, R, q')$, valuations $v \in r$, $\delta \in \mathbb{R}_{\geq 0}$ and $v' \in r'$ with $(q, v) \xrightarrow{\delta, t} (q', v')$. The initial state of the region automaton is $(q_0, [\mathbf{0}]_{\sim_{M_{\mathcal{A}}}})$ where $[\mathbf{0}]_{\sim_{M_{\mathcal{A}}}}$

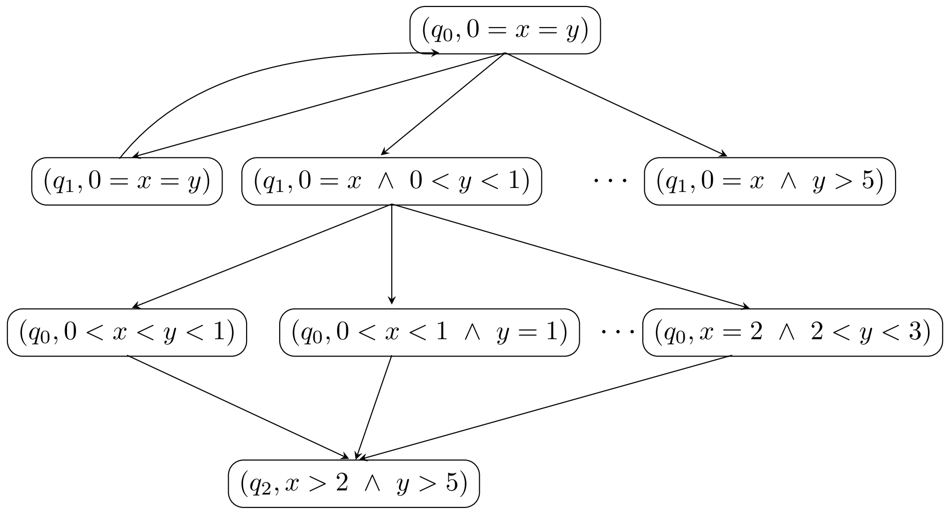

$(q_0, 0 = x = y)$
$(q_1, 0 = x = y)$
$(q_1, 0 = x \ \wedge \ 0 < y < 1)$
$\cdots$
$(q_1, 0 = x \ \wedge \ y > 5)$
$(q_0, 0 < x < y < 1)$
$(q_0, 0 < x < 1 \ \wedge \ y = 1)$
$\cdots$
$(q_0, x = 2 \ \wedge \ 2 < y < 3)$
$(q_2, x > 2 \ \wedge \ y > 5)$

Figure 1.4: Part of region automaton of the timed automaton $\mathcal{A}_1$ shown in Figure 1.1

represents the region to which the initial valuation $\mathbf{0}$ belongs to. A state (q, r) is accepting in the region automaton if q is an accepting state of $\mathcal{A}$. We denote the region automaton of $\mathcal{A}$ as $RA(\mathcal{A})$.

Figure 1.4 shows a part of the region automaton $RA(\mathcal{A}_1)$ of the automaton $\mathcal{A}_1$ shown in Figure 1.1.

It will be important to understand next the property of regions, similar to Proposition 14 shown for neighbourhoods. This property has been called pre-stability of regions [**?**].

Lemma 25 (Pre-stability of regions) Let $\mathcal{A}$ be an automaton. Transitions in $RA(\mathcal{A})$ are pre-stable: in each transition $(q, r) \xrightarrow{a} (q', r')$, for every $v \in r$ there is a $\delta \in \mathbb{R}_{\geq 0}$ and a valuation $v' \in r'$ such that $(q, v) \xrightarrow{\delta, a} (q', v')$

Proof
By definition of the region graph, a transition $(q_1, r_1) \xrightarrow{a} (q_2, r_2)$ exists in $RA(\mathcal{A})$ if there are $v_1 \in r_1$, $\delta \in \mathbb{R}_{\geq 0}$ and $v_2 \in r_2$ with $(q_1, v_1) \xrightarrow{\delta, a} (q_2, v_2)$.

Let the corresponding transition be (q_1, a, g, R, q_2). Pick a valuation $v_1' \in r_1$. By Lemma 22, there exists a δ' such that $v_1 + \delta$ and $v_1' + \delta'$ belong to the same region. We know that valuations within the same region satisfy the same guards. Therefore since $v_1 + \delta \vDash g$, we get that $v_1' + \delta' \vDash g$ too. From the definition of region equivalence, we get that regions are stable under projection to a subset of clocks and in particular, this entails that $[R](v_1' + \delta')$ belongs to the same region as $[R](v_1 + \delta)$. $\square$

We will now establish the correspondence between paths of the region graph and runs of the automaton. Consider two sequences

$$(q_0, v_0) \xrightarrow{\delta_0, a_0} (q_1, v_1) \xrightarrow{\delta_1, a_1} \cdots (q_n, v_n) \tag{1.1}$$

$$(q_0, r_0) \xrightarrow{a_0} (q_1, r_1) \xrightarrow{a_1} \cdots (q_n, r_n) \tag{1.2}$$

where the first is a run in $\mathcal{A}$, and the second is a path in $RA(\mathcal{A})$. We say that the first is an *instantiation* of the second if $v_i \in r_i$ for all $i \in \{1, \ldots, n\}$. Equivalently, we say that the second is an *abstraction* of the first. The following proposition is a direct consequence of the pre-stability property (similar in spirit to the Propositions 16 and 17).

Proposition 26 Every path in $RA(\mathcal{A})$ is an abstraction of a run of $\mathcal{A}$, and conversely, every run of $\mathcal{A}$ is an instantiation of a path in $RA(\mathcal{A})$.

The above lemma shows that the region graph is sound and complete for state reachability.

Theorem 27 ([AD94]) *Automaton $\mathcal{A}$ has an accepting run iff there is a path in the region graph $RA(\mathcal{A})$ starting from its initial node to an accepting node.*

The above theorem also gives an algorithm for solving the emptiness problem for timed automata: given a timed automaton, construct its region automaton and check for its emptiness. However this method is impractical. The number of regions obtained using a bound function M is $\mathcal{O}\big(|X|! \cdot 2^{|X|} \cdot \prod_{x \in X}(2M_x + 2)\big)$ [AD94] and constructing all of them, or even searching through them on-the-fly, has proved to be very costly. Later during the course, we will look at more efficient solutions to this problem.

References

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.