8 The task-graph example

8.1 Description of the problem

Compute \( D \times (C \times (A+B)) + (A+B) + (C \times D) \) using two processors:

- \( P_1 \) (fast):
  - Time: 2 picoseconds, 3 picoseconds
  - Energy:
    - Idle: 10 Watt
    - In use: 90 Watts

- \( P_2 \) (slow):
  - Time: 5 picoseconds, 7 picoseconds
  - Energy:
    - Idle: 20 Watts
    - In use: 30 Watts

8.2 Modelization of the problem

- Processors

8.3 How to model more?

How should we take into account:
- uncertainties on delays? (not mentioned in the lecture)
- power consumption?
- job preemption?
9 Hybrid systems

Hybrid systems are convenient models for representing general continuous behaviours. They have application in control theory, biology, etc.

Example 13. We want to model a thermostat. If $T$ is the temperature, it can written as a list of rules:

- when the heater is Off, the room cools down: $\dot{T} = -0.5T$;
- when the heater is On, the room heats: $\dot{T} = 2.25 - 0.5T$;
- when $21 \leq T \leq 22$, it switches Off;
- when $18 \leq T \leq 19$, it switches On.

This thermostat can be modelled by the following automaton:

A possible behaviour for that system is as follows:

A behaviour can also be seen as a sequence of delays and jumps.

Theorem 6. The reachability problem is undecidable in hybrid automata.

Therefore, to be able to analyze such systems, one needs to consider subclasses:

- dimension (number of variables), continuous- or discrete time
- what kind of dynamics
- what kind of guards, invariants, jumps

A zoology of subclasses:

- Linear hybrid automata: dynamics described by $\dot{x} = c$, polyhedral guards and invariants, linear resets
  - memory cell: $\dot{x} = 0$ (but $x$ can be reset)
  - clock: $\dot{x} = 1$
• stopwatch: in some locations, $\dot{x} = 1$, in others $\dot{x} = 0$
• skewed clock: $\dot{x} = c$ in all locations
  – Rectangular hybrid automata: dynamics described by $\dot{x} \in [c, d]$, guards and invariants described by $x \in [c, d]$, resets described by $x \in [c, d]$
  – Initialized rectangular hybrid automata: each time a variable changes its slope, it should be reset
  – Linear hybrid automata are rectangular hybrid automata
  – O-minimal hybrid automata: strong resets between locations, everything described in an o-minimal theory
  – Piecewise affine maps
  – Piecewise constant derivatives (PCD):
  – Polygonal differential inclusion systems (PDIS):
  – etc...

Exercise 8. Back to the task-graph scheduling problem, how would you model the power consumption? And how would you model job preemption?

In this lecture we will study:

1. Piecewise affine maps: rich jumps, poor dynamics
2. (Initialized) Rectangular hybrid automata: decorrelated variables (rather poor jumps)
3. Weighted timed automata, a timed-automata based model with observer variables

9.1 Piecewise affine maps (PAMs)

(Pictures borrowed from Eugene Asarin)

A $d$-dimensional piecewise affine map (PAM) is a discrete-time dynamical system defined as a tuple $(\mathcal{P}, (P_i)_{1 \leq i \leq n}, f)$ where:

– $\mathcal{P}$ is a bounded polyhedron of $\mathbb{R}^d$ (e.g. $\mathcal{P} = [0, 1]^d$);
– $(P_i)_{1 \leq i \leq n}$ is a polyhedral partition of $\mathcal{P}$;
– $f : \mathcal{P} \to \mathcal{P}$ is such that for every $1 \leq i \leq n$, $f|_{P_i}$ is affine.

The semantics of a PAM is given as a sequence $(f^i(x))_{i \geq 0}$, where $x \in \mathcal{P}$.

The reachability problem asks whether, given two points $x_1, x_2 \in \mathcal{P}$, there exists an index $n$ such that $f^n(x_1) = x_2$.

Theorem 7 ([KCG94]). Two-dimensional PAMs are undecidable.
Proof. Note that we cannot directly express a counter machine using a PAM since in a PAM, the domain is bounded.

However we can simulate a deterministic two-counter machine as follows: A state \((q_i, c, d)\) will be represented by point \((i - 1 + 2^{-c}, 2^{-d})\). The domain is therefore \((0, N] \times (0, 1]\) where \(N\) is the number of states of the two-counter machine. State \(q_i\) is encoded by the polyhedron \(P_i = (i - 1, i] \times (0, 1]\) (we may have to split this polyhedron for defining the affine functions). We now define the affine functions, depending on the instruction starting at \(q_i\):

- if \(q_i : c + +; \text{goto } q_j\), then we should go from point \((i - 1 + 2^{-c}, 2^{-d})\) to point \((j - 1 + 2^{-(c - 1)}, 2^{-d})\), that is \(y := y\) and \(x := j - 1 + \frac{2^{-(i - 1)}}{2}\).
- if \(q_i : c - -; \text{goto } q_j\), then we should go from point \((i - 1 + 2^{-c}, 2^{-d})\) to point \((j - 1 + 2^{-c - 1}, 2^{-d})\) in case \(c > 0\), that is from \(P_i \cap (x < i)\). The affine function is then defined as: \(x := j - 1 + 2(x - (i - 1))\) and \(y := y\).
- if \(q_i : \text{if } c > 0 \text{ then goto } q_j \text{ else goto } q_k\), then we should go from \(P_i \cap (x < i)\) to \(P_j\) using \(x := j - i + x\) and \(y := y\) and from \(P_j \cap (x = i)\) to \(P_k\) using \(x := k - i + x\) and \(y := y\).
- if \(q_i : d + +; \text{goto } q_j\), then we should go from point \((i - 1 + 2^{-c}, 2^{-d})\) to point \((j - 1 + 2^{-c}, 2^{-d - 1})\), that is \(x := x + j - i\) and \(y := \frac{y}{2}\).
- if \(q_i : d - -; \text{goto } q_j\), then we should go from point \((i - 1 + 2^{-c}, 2^{-d})\) to point \((j - 1 + 2^{-c}, 2^{-d + 1})\) on \(P_i \cap (y < 1)\), that is \(x := x + j - i\) and \(y := 2y\).
- if \(q_i : \text{if } d > 0 \text{ then goto } q_j \text{ else goto } q_k\), then we should go from \(P_i \cap y < 1\) to \(P_j\) using \(x := j - i + x\) and \(y := y\), and from \(P_j \cap y = 1\) to \(P_k\) using \(x := k - i + x\) and \(y := y\).

The encoding is illustrated on the following picture. Blue dots represent counter values.

If \(q_n\) is the halting state, and \(q_1\) is the initial state of the counter machine, the two-counter machine halts iff there is a path in the PAM from \((1, 1)\) (all counters set to 0 in \(q_1\)) to the area \((n - 1, n] \times (0, 1]\). By adding instructions decreasing the two counters when reaching \(q_n\), one can reduce to checking whether there is a path in the PAM from \((1, 1)\) to \((n, 1)\). \(\square\)

Remark 11 (An example open problem). It is unknown whether we can decide the reachability problem in one-dimensional PAMs.
9.2 Rectangular hybrid automata

We write $\mathcal{I}$ for the set of intervals of $\mathbb{R}$ with rational bounds.

A rectangular (hybrid) automaton is a tuple $\mathcal{H} = (L, \ell_0, X, E, \text{Inv}, \text{Act}, \text{Pre}, \text{Post}, \text{Upd})$ where:

- $L$ is a finite set of locations;
- $\ell_0 \in L$ is the initial location;
- $X = \{x_1, \ldots, x_n\}$ is a finite set of variables;
- $E \subseteq L \times L$ is a finite set of edges, each edge $e = (\ell, \ell') \in E$ is characterized by its source, $\text{src}(e) = \ell \in L$, and its target $\text{target}(e) = \ell' \in L$;
- $\text{Inv}: L \to \mathcal{I}^X$ assigns an invariant to every location;
- $\text{Act}: L \to \mathcal{I}^X$ indicates the evolution (aka activity) of the variables in each location;
- $\text{Pre}: E \to \mathcal{I}^X$ and $\text{Post}: E \to \mathcal{I}^X$ are the pre- and post-conditions to be fulfilled when firing transitions;
- $\text{Upd}: E \to 2^X$ indicates which variables should be updated.

It is said initialized whenever for every $e \in E$ for every $x \in X$, either $x \in \text{Upd}(e)$, or $\text{Act}(\text{src}(e))(x) = \text{Act}(\text{target}(e))(x)$.

We assume $\mathcal{H}$ is equipped with an extra variable $t$ which is a clock.

The semantics of $\mathcal{H}$ is given as a timed transition system $(S, s_0, \to)$ defined by $S = \{(\ell, \nu) \in L \times \mathbb{R}^X \mid \forall x \in X, \nu(x) \in \text{Inv}(\ell)(x)\}$, $s_0 = (\ell_0, 0)$, and $\to$ is defined as follows: $(\ell, \nu) \to (\ell', \nu')$ iff one of the two following conditions are met:

- [continuous step] $\ell = \ell'$, and either $\nu = \nu'$, or $\nu'(t) > \nu(t)$, and for every $x \in X$,
  \[
  \frac{\nu'(x) - \nu(x)}{\nu'(t) - \nu(t)} \in \text{Act}(\ell)(x)
  \]

- [discrete step (or jump)] there is a transition $e \in E$ such that $\text{src}(e) = \ell$, $\text{target}(e) = \ell'$, $\nu \models \text{Pre}(e)$, $\nu' \models \text{Post}(e)$, $\nu'(t) = \nu(t)$, for all $x \in X$, $\nu'(x) = \nu(x)$ unless $x \in \text{Upd}(e)$.

Example 14. The following is an example of an initialized rectangular automaton:

![Diagram of initialized rectangular automaton]

Remark 12. Obviously any timed automaton is an initialized rectangular automaton.
Undecidability of rectangular automata. A rectangular automaton is said simple whenever:

- only one variable, say $z$, is not a clock;
- the activity of $z$ is singular in each location;
- $\text{Inv}, \text{Pre}, \text{Post}$ have compact values;
- $\text{Post}(e)(x) = [0, 0]$ if $x \in \text{Upd}(e)$, and $\text{Post}(e)(x) = \text{Pre}(e)(x)$ otherwise.

Such a simple automaton is said 2-slope whenever the only non-clock variable can take two values only.

**Theorem 8.** The reachability problem is undecidable for simple 2-slope rectangular automata.

**Proof.** The result holds for any two different rationals serving as the slopes of the 2-slope variable, but we only prove it for the special case where the slopes are 0 and 1.

We begin with the case of stopwatches. We encode the halting problem for two-counter machines. Counters $c_1$ and $c_2$ are encoded by clocks $x_1$ and $x_2$, with $x_i = 2^{-c_i}$. We now have to be able to double and halve the value of a clock, preserving the value of the other clock at the same time. Doubling the value of a clock is achieved by the automaton depicted below:

Notice that, in this figure, we omit invariants in each state (but they can be easily inferred). Then it can be checked that the value of $c_1$ is doubled between the entry and exit of this automaton, while the value of $c_2$ is preserved. Halving the clock is achieved in a similar way, which can be done as an exercise.

**Exercise 9.** Show that we can decide the reachability problem for stopwatch automata with only two variables (we assume only resets to zero for the variables and rectangular guards). What happens if we allow diagonal constraints? Show the undecidability for three stopwatches and diagonal constraints.

Decidability of initialized rectangular automata.

**Theorem 9.** The reachability problem is decidable in initialized rectangular automata. It is even \textit{PSPACE}-complete.

**Proof.** We assume $\mathcal{H}$ is a rectangular automaton. W.l.o.g. we assume that:

- $\text{Pre}(e) \subseteq \text{Inv}(\text{src}(e))$;
- $\text{Post}(e) \subseteq \text{Inv}(\text{target}(e))$;

\begin{align*}
\ell_1 & : \{y,z\} \\
\ell_2 & : \{x_1, z\} \\
\ell_3 & : \{y\} \\
\ell_4 & : \{x_1\} \\
\ell_5 & : \{y\} \\
\ell_6 & : \{x_2\}
\end{align*}

Notice that, in this figure, we omit invariants in each state (but they can be easily inferred). Then it can be checked that the value of $c_1$ is doubled between the entry and exit of this automaton, while the value of $c_2$ is preserved. Halving the clock is achieved in a similar way, which can be done as an exercise.
Obviously, this restriction can be relaxed because of the above assumptions). We furthermore assume that all intervals are compact. W.l.o.g. we assume that \( \text{Inv} \) is always true (this does not affect reachability properties, because of the above assumptions). We have that:

- its set of variables is \( X' = \{ x^- , x^+ \mid x \in X \} \);
- variables are defined by \( \text{Act}'(\ell)(x^+) = [b, b] \) and \( \text{Act}'(\ell)(x^-) = [a, a] \) whenever \( \text{Act}(\ell)(x) = [a, b] \);
- We will have several copies of every edge in \( \mathcal{H} \): \( E' = E \times 4^X \). They are defined as follows.
  - Assume \( x \in \text{Upd}(e) \). Then any transition \( e' = (e, \sigma) \in E' \) will update \( x^- \) and \( x^+ \) \((\{x^- , x^+ \} \subseteq \text{Upd}'(e))\). Furthermore:
    \[
    \begin{align*}
    \text{Pre}(e)(x) = [a, b] & \implies \begin{cases} 
    \text{Pre}'(e^-)(x^-) = (-\infty, b] \\
    \text{Pre}'(e^+)(x^+) = [a, +\infty) 
    \end{cases} \\
    \text{Post}(e)(x) = [m, M] & \implies \begin{cases} 
    \text{Post}'(e^-)(x^-) = [m, m] \\
    \text{Post}'(e^+)(x^+) = [M, M] 
    \end{cases}
    \end{align*}
    \]
  - Assume \( x \notin \text{Upd}(e) \). We write \( \text{Pre}(e)(x) = [a, b] \) and \( \text{Post}(e)(x) = [m, M] \). Note that by hypothesis, \( [a, b] = [m, M] \). These constraints on \( x \) must be transferred to \( x^- \) and \( x^+ \). The way it will be handled depends on the relative positions of \( x^- \) and \( x^+ \) with regards to constants \( a \) and \( b \).

\begin{figure}[h]
\centering
\begin{subfigure}{0.2\textwidth}
\centering
\includegraphics[width=1\textwidth]{case1.png}
\caption{Case 1}
\end{subfigure}
\hfill
\begin{subfigure}{0.2\textwidth}
\centering
\includegraphics[width=1\textwidth]{case2.png}
\caption{Case 2}
\end{subfigure}
\hfill
\begin{subfigure}{0.2\textwidth}
\centering
\includegraphics[width=1\textwidth]{case3.png}
\caption{Case 3}
\end{subfigure}
\hfill
\begin{subfigure}{0.2\textwidth}
\centering
\includegraphics[width=1\textwidth]{case4.png}
\caption{Case 4}
\end{subfigure}
\end{figure}

\begin{itemize}
  \item **Case 1:** We fix \( \sigma \) such that \( \sigma(x) = 1 \), and define \( e' = (e, \sigma) \in E' \).
    \[
    \begin{align*}
    \begin{cases} 
    \text{Pre}'(e^-)(x^-) = [a, b] \\
    \text{Pre}'(e^+)(x^+) = [a, b] 
    \end{cases} & \implies \begin{cases} 
    \text{Post}'(e^-)(x^-) = [a, b] \\
    \text{Post}'(e^+)(x^+) = [a, b] 
    \end{cases}
    \end{align*}
    \]
    and variables \( x^- \) and \( x^+ \) are unchanged: \( x^-, x^+ \notin \text{Upd}'(e') \).
  \item **Case 2:** We fix \( \sigma \) such that \( \sigma(x) = 2 \), and define \( e' = (e, \sigma) \in E' \).
    \[
    \begin{align*}
    \begin{cases} 
    \text{Pre}'(e^-)(x^-) = [a, b] \\
    \text{Pre}'(e^+)(x^+) = (b, +\infty) 
    \end{cases} & \implies \begin{cases} 
    \text{Post}'(e^-)(x^-) = [a, b] \\
    \text{Post}'(e^+)(x^+) = [b, b] 
    \end{cases}
    \end{align*}
    \]
    variable \( x^- \) is unchanged whereas variable \( x^+ \) is changed: \( x^- \notin \text{Upd}'(e') \) but \( x^+ \in \text{Upd}'(e') \).
\end{itemize}
* **Case 3**: We fix $\sigma$ such that $\sigma(x) = 3$, and define $e' = (e, \sigma) \in E'$.

\[
\begin{aligned}
\text{Pre}'(e')(x^-) &= (-\infty, a) \\
\text{Post}'(e')(x^-) &= [a, a] \\
\text{Pre}'(e')(x^+) &= [a, b] \\
\text{Post}'(e')(x^+) &= [a, b]
\end{aligned}
\]

variable $x^+$ is unchanged whereas variable $x^-$ is changed: $x^+ \notin \text{Upd}'(e')$ but $x^- \in \text{Upd}'(e')$.

* **Case 4**: We fix $\sigma$ such that $\sigma(x) = 4$, and define $e' = (e, \sigma) \in E'$.

\[
\begin{aligned}
\text{Pre}'(e')(x^-) &= (-\infty, a) \\
\text{Post}'(e')(x^-) &= [a, a] \\
\text{Pre}'(e')(x^+) &= (b, +\infty) \\
\text{Post}'(e')(x^+) &= [b, b]
\end{aligned}
\]

variables $x^+$ and $x^-$ are changed: $\{x^-, x^+\} \subseteq \text{Upd}'(e')$.

**Example 15.** The transformation applied to the previous example yields:

\[
\begin{aligned}
\dot{x}^+ &= 2 \\
\dot{x}^- &= 1 \\
x^+ &\geq 2 \\
x^- &\leq 5
\end{aligned}
\]

\[
\begin{aligned}
\dot{x}^+ &= 3 \\
\dot{x}^- &= 0 \\
x^+ &:= 2, x^- := 1
\end{aligned}
\]

\[
\begin{aligned}
\dot{x}^+ &= 3 \\
\dot{x}^- &= 0 \\
x^- &:= 3, x^+ := 4
\end{aligned}
\]

\[
\begin{aligned}
\dot{x}^+ &\leq 3 \\
x^- &\geq 3 \\
x^+ &\leq 4 \\
x^- &\leq 4 \\
x^+ &\leq 4
\end{aligned}
\]

\[
\begin{aligned}
3 &\leq x^- \wedge x^+ \leq 4 \\
3 &\leq x^- \leq 4 \\
x^- &\leq 4 \\
x^+ &\leq 4
\end{aligned}
\]

We now discuss the correctness of the construction of $\mathcal{M}$. For every configuration $(\ell, \nu)$ of $\mathcal{M}$ we define the following set of configurations of $\mathcal{H}$:

\[
\chi(\ell, \nu) = \{ (\ell, v) \mid \forall x \in X, \nu(x^-) \leq v(x) \leq \nu(x^+) \}
\]

We define the following set of states of $\mathcal{M}$: $U = \{ (\ell, \nu) \mid \forall x \in X, \nu(x^-) \leq \nu(x^+) \}$.

Obviously, $\chi(\ell, \nu) \neq \emptyset$ iff $(\ell, \nu) \in U$.

We can easily prove:

**Lemma 10.** Let $(\ell, \nu) \in U$, $t \in \mathbb{R}_+$ and $e \in E$. Then:

- $\text{timePost}^t_H(\chi(\ell, \nu)) = \chi(\text{timePost}^t_M(\ell, \nu))$;
- $\text{discPost}^e_H(\chi(\ell, \nu)) = \chi(\text{discPost}^e_M(\ell, \nu))$.

where $\text{timePost}^t$ is the time-successor after a delay of $t$ time units, and $\text{discPost}^e$ is the discrete successor after edge $e$.

As a consequence, the set of reachable states in $\mathcal{H}$ is the image by $\chi$ of the set of reachable states in $\mathcal{M}$.

**Remark 13.** Let us summarize the characteristics of multi-rate automaton $\mathcal{M}$:
– for each variable \( y \), for each location \( \ell \), \( \text{Act}(\ell)(y) \) is a singleton;
– for every edge \( e \), if \( y \in \text{Upd}(e) \), then \( \text{Post}(e)(y) \) is a singleton;
– for every edge \( e \), if \( y \notin \text{Upd}(e) \), then \( \text{Pre}(e)(y) = \text{Post}(e)(y) \).

Notice also that if \( \mathcal{H} \) is initialized, then so is \( \mathcal{M} \).

We now transform the (initialized) multi-rate automaton \( \mathcal{M} \) into an initialized stopwatch automaton\(^{21} \) \( \mathcal{S} \). The idea is to scale the variables by a correct factor. We assume the set of variable of \( \mathcal{M} \) to \( \alpha \) define \( m \) watch automaton.

For every set of states \( \mathcal{S} \) initialized. We let the number of states of the automaton. We thus get a \( \text{PSPACE} \) diagonal-free timed automata.

However, notice that this yields an exponential blowup in the size of the system. Therefore, those are harmless in the standard region automata construction can be used to prove decidability of this class of systems (adapt the compatibility of the regions with resets).

Exercise 10. Show that we can remove updates of the form \( x := c \) in timed automata. However, notice that this yields an exponential blowup in the size of the system. Therefore, show that standard region automata construction can be used to prove decidability of this class of systems (adapt the compatibility of the regions with resets).

\(^{21}\) That is, in each location, each variable has either slope 0 or 1.
9.3 Weighted timed automata

**Weighted timed automaton.** A *weighted timed automaton* is a timed automaton with an observer weight variable. It is defined as a tuple $A = (L, \ell_0, L_F, X, \Sigma, T, \text{weight})$, where:

- $(L, \ell_0, L_F, X, \Sigma, T)$ is a standard (diagonal-free) timed automaton, and
- $\text{weight} : L \cup T \to \mathbb{Z}$ assigns a value to each location and to each transition.

We assume $\Sigma = T$, that is each transition can be identified with its label.

The weight of a delay move $(\ell, v) \xrightarrow{d} (\ell, v + d)$ is given by $d \cdot \text{weight}(\ell)$: $\text{weight}(\ell)$ is the rate in $\ell$. The weight of a discrete move $(\ell, v) \xrightarrow{e} (\ell', v')$ is given by $\text{weight}(e)$. The weight of a finite run $\varrho$ is the accumulated weight of all moves that compose the run $\varrho$, we write $\text{cost}(\varrho)$ for that value, and we call it the cost of $\varrho$.

**Example 16.** We consider the following weighted timed automaton:

A possible execution for that system is:

<table>
<thead>
<tr>
<th></th>
<th>$\ell_0$</th>
<th>$\ell_0$</th>
<th>$\ell_1$</th>
<th>$\ell_1$</th>
<th>$\ell_3$</th>
<th>$\ell_3$</th>
<th>$\ell_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>0</td>
<td>1.3</td>
<td>1.3</td>
<td>1.3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y$</td>
<td>0</td>
<td>1.3</td>
<td>0</td>
<td>0</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>cost</strong></td>
<td>6.5</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>7</td>
</tr>
</tbody>
</table>

**Remark 14.** Note that weighted timed automata can be seen as linear hybrid automata, or as rectangular automata. However the weight variable is never constrained, that is, the behaviour of the weighted timed automaton is that of the underlying timed automaton.

**Example 17.** Back to the taskgraph scheduling problem, we can refine the models for the processors and get:
Optimal reachability in weighted timed automata

We first focus on the optimal reachability question: given a weighted timed automaton $A$, can we compute the optimal cost that allows to reach the set of target states. The optimal cost is defined as

$$\text{optcost}_A = \inf \{ \text{cost}(\varrho) \mid \varrho \text{ run from } (\ell_0, 0) \text{ to } L_F \}$$

**Example 18.** If we consider the weighted timed automaton of Example 16, this optimal cost can be computed as:

$$\inf_{0 \leq t \leq 2} \left( 5t + \min(10(2 - t) + 1, (2 - t) + 7) \right) = 9$$

and the “strategy” is to take the first transition when $x = 0$.

**Theorem 10.** We can compute the optimal cost for reaching the target location in weighted timed automata in polynomial space.

First notice that the region abstraction is too rough to correctly take into account the cost information:

![Diagram of region abstraction](image)

Therefore the proof of this theorem relies on a refinement of the region automaton construction, called the *corner-point abstraction*, which is a finite weighted graph:

![Diagram of corner-point abstraction](image)

We fix a weighted timed automaton $A$, and w.l.o.g. we assume all clocks are bounded by maximal constant $M$. We fix its set of diagonal-free regions $R$. A corner-point is a pair $(r, \alpha)$ where $r \in R$, and $\alpha \in \mathbb{F}$ with integral coordinates. Note that if $r$ is given by the following order on fractional part $X_0 < X_1 < \cdots < X_p$ where $(X_i)$ is a partition of the set of clocks, $X_0$ contains all clocks with integral values, for every $i$, the fractional parts of all clocks in $X_i$ are equal, and all clocks in $X_i$ have fractional part strictly larger than clocks
in $X_{i-1}$, then there are $p + 1$ corner-points to $r$, each one is characterized by $0 \leq i \leq p$ such that the value of clocks in $X_0 \cup X_1 \cup \cdots \cup X_i$ is the integral part whereas the value of clocks in $X_{i+1} \cup \cdots \cup X_p$ is the integral part plus one.

We write $R_{cp}$ for the set $\{(r, \alpha) \mid r \in R$ and $\alpha$ is a corner-point of $r\}$. We write $r_0$ for the region where all clocks have value 0, and $\alpha_0$ for its unique corner-point.

**Example 19.** Region $0 < x < 1 \land 1 < y < 2 \land y - x < 1$ has three corner-points:

$$(x = 0, y = 1) \quad (x = 1, y = 1) \quad (x = 0, y = 2)$$

The corner-point abstraction $R_{cp}(A)$ of $A$ is a finite weighted graph $(Q, q_0, Q_F, \rightarrow)$ where $Q = L \times R_{cp}$, $q_0 = (\ell_0, r_0, \alpha_0)$, $Q_F = L \times R_{cp}$, and $\rightarrow \subseteq Q \times Z \times Q$ is defined as follows:

1. there is a transition $(\ell, r, \alpha) \xrightarrow{\text{weight}(\ell)} (\ell, r, \alpha')$ if $\alpha$ and $\alpha'$ are corner-points of $r$ (this is only possible if $\alpha' = \alpha + 1$)
2. there is a transition $(\ell, r, \alpha) \xrightarrow{0} (\ell, r', \alpha)$ if $r'$ is the immediate successor of $r$, and $\alpha$ is a corner-point of both $r$ and $r'$
3. there is a transition $(\ell, r, \alpha) \xrightarrow{\text{weight}(e)} (\ell', r', \alpha')$ if edge $e = (\ell, g, Y, \ell')$ is such that $r \subseteq [g]$, $r' = [Y \leftarrow 0]r$ and $\alpha' = [Y \leftarrow 0]\alpha\text{.}^{22}$

We will show that this abstraction is sound and complete w.r.t. optimal reachability, that is $\text{optcost}_A = \text{optcost}_{R_{cp}(A)}$. More precisely we will prove the following proposition:

**Proposition 9.** For every accepted finite run $\varrho$ in $A$, there is an accepted finite path $\pi$ in $R_{cp}(A)$ such that $\text{cost}(\pi) \leq \text{cost}(\varrho)$.

For every accepted finite path $\pi$ in $R_{cp}(A)$, for every $\varepsilon > 0$, there is an accepted finite run $\varrho$ in $A$ such that $\text{cost}(\varrho) \leq \text{cost}(\pi) + \varepsilon$.

We first give some technical results.

This section contains technical results that will be useful in the following. Let $A$ be a closed set of $\mathbb{R}^n$ (with $n \geq 1$). The border of $A$ is denoted by $\text{Border}_n(A)$ and is defined as $A \setminus \hat{A}$ where $\hat{A}$ denotes the interior of $A$. Let $A$ be a closed set and $x$ a point in $\mathbb{R}^n$. The following statements are equivalent and characterize the border of $A$:

1. $x \in \text{Border}_n(A)$
2. $x \in A$ and for every $\varepsilon > 0$, there exists $y \not\in A$ such that $\|x - y\|_\infty < \varepsilon\text{.}^{23}$

**Lemma 12.** Let $n \geq 1$ and $Z$ be a bounded zone. Let $f$ be a function

$$f : (x_1, \ldots, x_n) \mapsto \sum_{i=1}^{n} c_i x_i + c$$

Then $\inf_{Z} f$ is obtained on the border of $\overline{Z}$ at integer coordinates.

---

22 Note that $[Y \leftarrow 0]r$ is a corner-point of $[Y \leftarrow 0]r$ if $\alpha$ is a corner-point of $r$.
23 $\| \cdot \|_\infty$ represents the usual infinite norm defined as $\|(x_i)_{i=1,\ldots,n}\|_\infty = \max\{|x_i| \mid i = 1,\ldots,n\}$.
Proof. The result is trivial when $n = 1$.

Assume now $n > 1$, and fix a compact set $A$. The minimum of $f$ on $A$ is obtained for some value $\alpha$ for $x_1$. Now, assume that $g_\alpha(x_2, ..., x_n) = f(\alpha, x_2, ..., x_n)$ ($\alpha$ is viewed as a parameter). Fix that $\alpha$, and notice that $A \cap (x_1 = \alpha) \neq \emptyset$. The function $g_\alpha$ is defined on a subset of $\mathbb{R}^{n-1}$. We denote by $B_\alpha$ the projection of $A \cap (x_1 = \alpha)$ onto the last $n - 1$ coordinates, $B_\alpha$ is a compact set of $\mathbb{R}^{n-1}$. Note also that $g_\alpha$ is also affine. We know from induction hypothesis that $\min_{B_\alpha} g_\alpha$ is obtained on $\text{Border}_{n-1}(B_\alpha)$. To get the induction step, it is then sufficient to prove that if $(x_2, ..., x_n)$ is in $\text{Border}_{n-1}(B_\alpha)$, then $(\alpha, x_2, ..., x_n)$ is in $\text{Border}_n(A)$.

Pick $(x_2, ..., x_n)$ in $\text{Border}_{n-1}(B_\alpha)$ and $\varepsilon > 0$. Then $(x_2, ..., x_n) \in B_\alpha$ and there exists $(y_2, ..., y_n) \notin B_\alpha$ such that $\|(x_2, ..., x_n) - (y_2, ..., y_n)\|_\infty < \varepsilon$. $B_\alpha$ is the projection of $A \cap (x_1 = \alpha)$ onto the $n - 1$ last coordinates, thus $(\alpha, x_2, ..., x_n) \in A \cap (x_1 = \alpha)$ and $(\alpha, y_2, ..., y_n) \notin A \cap (x_1 = \alpha)$. However, $\|(\alpha, x_2, ..., x_n) - (\alpha, y_2, ..., y_n)\|_\infty = \|(x_2, ..., x_n) - (y_2, ..., y_n)\|_\infty < \varepsilon$. We conclude that $(\alpha, x_2, ..., x_n)$ is in $\text{Border}_n(A)$.

Consider now the case where $f$ is defined on a bounded zone $Z$. Applying the previous result, we get that $\min_Z f$ is obtained on $\text{Border}_n(Z)$. Recall that $Z$ can be obtained from $Z$ by replacing the constraints $x - y < c$ by $x - y \leq c$ and that $\text{Border}_n(Z)$ corresponds to the union of all the facets of $Z$.\footnote{A facet of a closed zone $Z$ is an intersection $Z \cap (x = c)$ (or $Z \cap (x - y = c)$) where $x \{\leq, \geq\} c$ (or $x - y \{\leq, \geq\} c$) is a constraint, as tight as possible, defining $Z$.}

The infimum of $f$ on $Z$ is thus on a facet whose equation is $x - y = c$ (resp. $x = c$). We eliminate variable $x$ in $f$ by replacing $x$ with $y + c$ (resp. $c$) and we get a function $g$ which has $n - 1$ variables. Without loss of generality we can assume that $x = x_1$. We then use the property that $\min_Z f = \min_{Z'} g$ where $Z' = \text{proj}_{(x_2, ..., x_n)}(Z \cap x_1 = y + c)$ (resp. $Z' = \text{proj}_{(x_2, ..., x_n)}(Z \cap x_1 = c)$).

We know by induction hypothesis that the minimum of $g$ is obtained with each $x_i$ ($i > 1$) having integer values. Thus, the minimum of $f$ is obtained with $x_1 = y + c$ (resp. $x_1 = c$) which is an integer and all other clocks also have integer values. \hfill \Box

This lemma implies that the infimum of such a function $f$ on a bounded zone $Z$ is obtained in one of the extremal points of the zone.

Proof (of Proposition 9). We first prove the correctness, and then the soundness.

Correctness. Let $\varrho = (\ell_0, u_0) \rightarrow (\ell_0, u_0 + d_0) \rightarrow (\ell_1, u_1) \rightarrow (\ell_1, u_1 + d_1) \cdots \rightarrow (\ell_n, u_n)$ be a finite run in $\mathcal{A}$ (with alternating delay and discrete transitions). We moreover assume that this execution is read on the sequence of transitions $\ell_0 \xrightarrow{g_1,Y_1} \ell_1 \cdots \xrightarrow{g_n,Y_n} \ell_n$ in $\mathcal{A}$. The cost of $\varrho$ is given by:

$$f(d_0, d_1, ..., d_{n-1}) = \sum_{i=0}^{n-1} c_i \cdot d_i + c$$

where $c_i$'s are the weights of the locations $\ell_i$'s, and $c$ is the sum of all the discrete weights of transitions along $\varrho$.
We want to minimize this function with the constraints that we have a run which is region-equivalent to $\varrho$:

- if $v_i(x) = \sum_{h=j}^{i-1} d_h$ where $j = \max\{k \leq i \mid x \in Y_k\}$ and $v'_i(x) = \sum_{h=j}^{i} d_h$ where $j = \max\{k \leq i \mid x \in Y_k\}$
- then $v_i \in r_i$, $v'_i \in r'_i$ where $r_i$ (resp. $r'_i$) is the region of $u_i$ (resp. $u_i + d_i$).

The (topological) closure of this set of constraints on variables $D = (d_i)_{0 \leq i \leq n-1}$ can be represented by a linear constraint $M \cdot D \leq A$, where each line of $M$ is of the form $0 \ldots 0 1 \ldots 1 0 \ldots 0$ or $0 \ldots 0 -1 \ldots -1 0 \ldots 0$. One can show that such a matrix is totally unimodular.\(^{25}\) This implies that minimizing $f$ along the region path of $g$ yields an integer solution $(\alpha_i)_{0 \leq i \leq n-1}$.

**Example 20 (Optimal reachability as a linear programming problem).** We illustrate the above construction:

\[ t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow \ldots \quad \begin{cases} t_2 \leq 2 \\ t_4 - t_1 \geq 5 \end{cases} \]

We define the valuations $(\sigma_i)_{0 \leq i \leq n}$ by $\sigma_i(x) = \sum_{h=j}^{i-1} \alpha_h$ where $j = \max\{k \leq i \mid x \in Y_k\}$. Each valuation $\sigma_i$ is in $\overline{r_i}$ and has integer coordinates. It is thus a corner-point of $r_i$. Moreover, the sequence of valuations $(\sigma_i)$ would be an accepted sequence if we replace the constraints $r_i$ by $\overline{r_i}$. In addition, the time elapsed in each state $\ell_i$ would then be $d_i$. It is technical but easy to build a corresponding path $\pi$ in the corner-point abstraction $R_{cp}(A)$.

As $(\alpha_i)_{i=1 \ldots n}$ minimizes $f$ over the closure of the constraint defined by the region path of $g$, we get that $\text{cost}(\pi) \leq \text{cost}(g)$ and we are done.

**Completeness.** Note that if all guards are closed, then this is straightforward. In the general case, it relies on the following lemma:

**Lemma 13.** Let $\pi = (\ell_0, r_0, \alpha_0) \rightarrow (\ell_1, r_1, \alpha_1) \rightarrow \ldots$ be a path in $A_{cp}$. For every $\varepsilon > 0$, there exists a real run $\varrho_\varepsilon = (\ell_0, v_0) \rightarrow (\ell_1, v_1) \rightarrow \ldots$ such that for every $i$, $v_i \in r_i$ and $\|\alpha_i - v_i\|_\infty < \varepsilon$.

To prove this lemma we show the following: for every $(\ell, r, \alpha) \rightarrow (\ell', r', \alpha')$, for every $\varepsilon > 0$, for every $v \in r$ such that $\delta_\alpha(v) < \varepsilon$, there exists $(\ell, v) \rightarrow (\ell', v')$ in $A$ such that $v' \in r'$ and $\delta_{\alpha'}(v') < \varepsilon$, where the diameter of $v$ w.r.t. $\alpha$ is defined as $\delta_\alpha(v) = \max\left(\max_x(|v(x) - \alpha_x|), \max_{x,y}(|(v(x) - \alpha_x) - (v(y) - \alpha_y)|)\right)$.

To prove this we distinguish between all cases for the transition $(\ell, r, \alpha) \rightarrow (\ell', r', \alpha')$.

- Assume $(\ell, r, \alpha) \rightarrow (\ell', r', \alpha')$ is a discrete transition and $\delta_\alpha(v) < \varepsilon$ with $v \in r$. Then let $v'$ be such that $(\ell, v) \rightarrow (\ell', v')$. Let $Y$ be the set of clocks which are reset along that move. Then $v'(y) = 0$ and $\alpha'_y = 0$ for every $y \in Y$, and $v'(y) = v(y)$ and $\alpha'_y = \alpha_y$ for every $y \notin Y$. Then obviously $\delta_{\alpha'}(v') \leq \delta_\alpha(v) < \varepsilon$.

\(^{25}\) Which means that every square matrix extracted from $M$ has determinant $-1$, 0 or 1.
Assume \((\ell, r, \alpha) \rightarrow (\ell', r', \alpha')\) is a delay transition. There are several cases:

- \((\ell, r, \alpha) \rightarrow (\ell, r, \alpha + 1)\)

  \[
  \begin{array}{c}
  \alpha \\
  v(x) v(z) v(y)
  \end{array}
  \quad
  \begin{array}{c}
  \alpha + 1
  \end{array}
  \quad
  \begin{array}{c}
  < \varepsilon
  \end{array}
  \]

  In that case, write \(\tau = \min\{v(x) - \alpha_x \mid x \in X\}\) and \(\tau' = \max\{v(x) - \alpha_x \mid x \in X\}\), and define \(v' = v + 1 - \tau' - \tau\). With this value we easily get \(\delta_{\alpha+1}(v') = \delta_{\alpha}(v) < \varepsilon\).

  The new situation is illustrated below:

  \[
  \begin{array}{c}
  \alpha \\
  v'(x) v'(z) v'(y)
  \end{array}
  \quad
  \begin{array}{c}
  < \varepsilon
  \end{array}
  \quad
  \begin{array}{c}
  \alpha + 1
  \end{array}
  \]

- \((\ell, r, \alpha) \rightarrow (\ell, r', \alpha)\) where \(r'\) is the immediate successor of \(r\). There are several cases, which depend on the position of \(v\) relative to \(\alpha\).

  * There are some clocks \(x\), such that \(v(x) < \alpha_x\), and there is no clock \(y\) such that \(v(y) = \alpha_y\). This case is illustrated below.

    \[
    \begin{array}{c}
    \alpha \\
    v(x) v(z) v(y)
    \end{array}
    \quad
    \begin{array}{c}
    < \varepsilon
    \end{array}
    \]

    In that case, we define \(\tau = \min\{\alpha_x - v(x) \mid v(x) < \alpha_x\}\), and then \(v' = v + \tau\). We get that \(v' \in r'\), and that \(\delta_{\alpha}(v') = \delta_{\alpha}(v) < \varepsilon\). The new situation is illustrated below.

    \[
    \begin{array}{c}
    \alpha \\
    v'(x) v'(z) v'(y)
    \end{array}
    \quad
    \begin{array}{c}
    < \varepsilon
    \end{array}
    \]

  * There is some clock \(x\) such that \(v(x) = \alpha_x\), and there is some clock \(y\) such that \(v(y) < \alpha_y\).

    \[
    \begin{array}{c}
    \alpha \\
    v(y) v(x) v(z)
    \end{array}
    \quad
    \begin{array}{c}
    < \varepsilon
    \end{array}
    \]

    We let \(\tau = \min\{\alpha_y - v(y) \mid v(y) < \alpha_y\}\), and we define \(v' = v + \frac{\tau}{2}\). We get the expected result, illustrated below.

    \[
    \begin{array}{c}
    \alpha \\
    v'(y) v'(x) v'(z)
    \end{array}
    \quad
    \begin{array}{c}
    < \varepsilon
    \end{array}
    \]

  * This is the “most complex” case. There is some clock \(x\) such that \(v(x) = \alpha_x\), and there is no clock \(y\) such that \(v(y) < \alpha_y\).
In that case, we let $\tau = \varepsilon - \delta_{\alpha}(v)$ (which we know is positive), and we let $v' = v + \frac{\tau}{2}$. We then get the expected result. $> \delta_{v}(\alpha)$ but $< \varepsilon$

This concludes the proof. \qed

Now, along a given sequence of transitions, the cost is continuous, which means that for every $\nu > 0$, we can find some $\varepsilon > 0$ such that the distance between two runs is smaller than $\varepsilon$ implies that the difference of costs for the two runs is smaller than $\nu$. This concludes the completeness of the construction.