Regions in SML

Kim G. Larsen & Paul Pettersson

1 Regions

In this more theoretical exercise you are asked to develop and implement a datastructure in SML, which allows regions to be represented and manipulated as required when constructing the region automata (see JPK defn. 54) of a timed automata. In dealing with the exercise you may find it useful to consult sections 4.6 and 4.7 in JPK.

First, we recall the definition of clock equivalence (JPK defn. 49). In the definition C is a given finite set of clocks and $V(C) = C \to \mathbb{R}^+$ is the set of clock valuations ranged over by v, v'. In our definition we make the simplifying assumption that there is a single maximal constant $c_{\max}$ valid for all clocks.

Definition 1 *Two clock valuations v and v' are equivalent ($v \approx v'$) provided the following three conditions hold:*

1. $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ *or* $v(x), v'(x) > c_{max}$*, for all* $x \in C$*, and*
2. $\text{frac}(v(x)) \leq \text{frac}(v(y))$ *iff* $\text{frac}(v'(x)) \leq \text{frac}(v'(y))$ *for all* $x, y \in C$ *with* $v(x), v(y) \leq c_{max}$*, and*
3. $\text{frac}(v(x)) = 0$ *iff* $\text{frac}(v'(x)) = 0$ *for all* $x \in C$ *with* $v(x) \leq c_{max}$*.*

A region γ *is an equivalence class of clock valuations with respect to* $\approx$[1]*.*

Figure 1 illustrates the 18 equivalence classes resulting for two clocks when the maximal constant is 1. In words, equivalent clock valuations must (essentially):

[1] Again a sligth difference compared with JPK, where a region is a pair (l, γ), where l is a location of the given timed automata and γ is an equivalence class of clock valuations with respece to $\approx$.

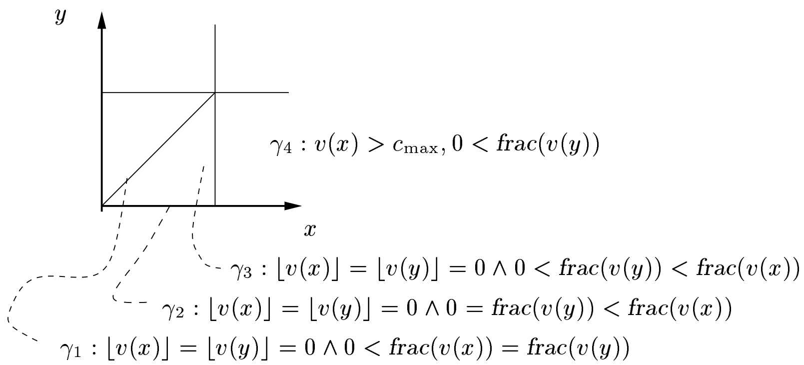

Figure 1: Some of the 18 regions when $C = \{x, y\}$ and $c_{\max} = 1$.

1. have same integral part or exceed $c_{\max}$ for all clocks;
2. agree on the clocks for which the fractional part is 0;
3. agree on the ordering of clocks determined by the size of the fractional part.

Due to this agreement, we may lift information of integral and functional parts from clock valuations to their regions (equivalence classes). In particular, we write:

- $\gamma(x) > c_{\max}$ when $v(x) > c_{\max}$ for all $v \in \gamma$,
- $\gamma(x) \leq c_{\max}$ when $v(x) \leq c_{\max}$ for all $v \in \gamma$.

For clocks x and y satisfying the latter condition we have in addition:

- $\lfloor \gamma(x) \rfloor = k$ when $\lfloor v(x) \rfloor = k$ for all $v \in \gamma$,
- $\mathit{frac}(\gamma(x)) = 0$ when $\mathit{frac}(v(x)) = 0$ for all $v \in \gamma$,
- $\mathit{frac}(\gamma(x)) \leq \mathit{frac}(\gamma(y))$ when $\mathit{frac}(v(x)) \leq \mathit{frac}(v(y))$ for all $v \in \gamma$,

Exercise 1 *Argue that if $v \approx v'$ then also*

1. $\mathit{frac}(v(x)) < \mathit{frac}(v(y))$ *iff* $\mathit{frac}(v'(x)) < \mathit{frac}(v'(y))$ *for all* $x, y \in C$ *with* $v(x), v(y) \leq c_{max}$*, and*
2. $\mathit{frac}(v(x)) = \mathit{frac}(v(y))$ *iff* $\mathit{frac}(v'(x)) = \mathit{frac}(v'(y))$ *for all* $x, y \in C$ *with* $v(x), v(y) \leq c_{max}$.

2 List Representation of Regions

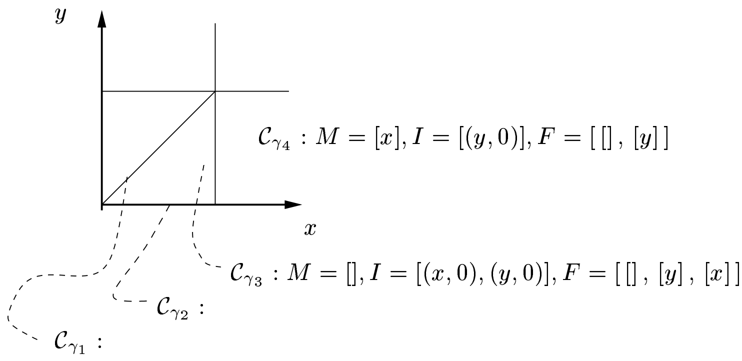

Figure 2: List representation of some of the regions of Figure 1.

In this section we suggest a list-based representation of regions. For A a set we denote by $\mathsf{List}(A)$ the set of all lists over A. For L a list over A and a an element of A we misuse notation slightly and denote by $a \in L$ that a occurs as an element in L.

A region γ is representated by a triple $\mathcal{C}_\gamma = (M_\gamma, I_\gamma, F_\gamma)$, where:

- $M_\gamma : \mathsf{List}(C)$,
- $I_\gamma : \mathsf{List}(C \times \mathbb{N})$,
- $F_\gamma : \mathsf{List}(\mathsf{List}(C))$.

The three lists of $\mathcal{C}_\gamma$ represent γ in the following way:

- M_γ contains all clocks $x \in C$ for which $\gamma(x) > c_{\max}$.
- I_γ gives information of the integral parts of the remaining clocks, i.e. $(x,k) \in I_\gamma$ iff $\lfloor \gamma(x) \rfloor = k$.
- The list F_γ should have the form $F_\gamma = [L_0, L_1, \ldots, L_k]$, where the lists $L_0, L_1, \ldots, L_k$ is a partitioning of the clocks x for which $\gamma(x) \leq c_{\max}$, i.e. each such clock belongs to precisely one list L_i. Only L_0 is allowed to be empty. Now, F_γ gives information of the relative ordering of fractional parts in a rather obviuos way. L_0 contains all clocks with fractional part equal to 0; clocks belonging to the same list, L_i, have

identical fractional parts; and the fractional parts of clocks increase according to the index of their enclosing list, L_i. That is:

- $\mathit{frac}(\gamma(x)) = 0$ iff $x \in L_0$,
- $\mathit{frac}(\gamma(x)) \leq \mathit{frac}(\gamma(y))$ iff $x \in L_i$ and $y \in L_j$ with $i \leq j$.

Exercise 2 *Give the list representations of the regions γ_1 and γ_2 in Figure 2.*

3 Implementation of Region Operations

In the following you are asked to define a number of useful operations on regions based on the list representation suggested in the previous section. Also, you are requested to implement the various operations as list-processing functions in SML[2]. Thus, let $\mathcal{C}_\gamma = (M_\gamma, I_\gamma, F_\gamma)$ be the list representation of some region γ over clocks C with maximal constant $c_{\max}$.

Guards

Define (and implement in SML) a boolean function $\texttt{sat}_{\leq}$ that given a clock x, a bound k and a list representation $\mathcal{C}_\gamma$ will test if γ satisfies the bound $x \leq k$. That is:

$$\texttt{sat}_{\leq}(x, k, \mathcal{C}_\gamma) = \texttt{true}$$

if and only if γ satisfies the guard $x \leq k$ (i.e. v satisfies $x \sim k$ for all $v \in \gamma$). Define in a similar way functions $\texttt{sat}_{\geq}, \texttt{sat}_{<}$ and $\texttt{sat}_{>}$. As an example $\texttt{sat}_{\leq}(x, 1, \mathcal{C}_{\gamma_3}) = \texttt{true}$ in Figure 2.

Reset

Define (and implement in SML) a function `reset` that given a clock x and a list representation of a regions γ returns the list representation of the region $\beta = (\mathsf{reset}\, x\, \mathsf{in}\, \gamma)$[3]. That is:

$$\texttt{reset}(x, \mathcal{C}_\gamma) = \mathcal{C}_\beta$$

As an example $\texttt{reset}(x, \mathcal{C}_{\gamma_3}) = \mathcal{C}_{\gamma_2}$ in Figure 2.

[2] or another functional language such as LISP or SCHEME.

[3] see JPK p 200

Delay-Successor

Define (and implement in SML) a function `succ` which given a list representation, $\mathcal{C}_\gamma$, returns a list representation, $\mathcal{C}_\delta$, of the delay-successor region, $\delta = \mathit{delsucc}(\gamma)^4$.

As an example $\mathtt{succ}(\mathcal{C}_{\gamma_2}) = \mathcal{C}_{\gamma_3}$ in Figure 2.

4 Application

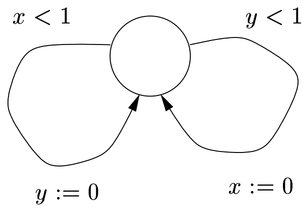

Figure 3: A simple timed automata

Consider the very simple timed automata of Figure 3. In this exercise you are asked to use the SML functions from the previous section to explore the corresponding region automata. Which of the following properties are reachable from the initial region (i.e. $x = 0 \wedge y = 0$):

- $x = 0 \wedge y < 1$
- $x = 1 \wedge y = 0$
- $y < 1 \wedge x > 1$
- $x > 1 \wedge y = 0$

In the affirmative cases provide a sequence of regions (preferably the shortest) that will bring the automata from the initial state/region to a region satisfying the property.

Give, using you SML implementation, a diagram showing the reachable (i.e. from the initial region) part of the region automata (a'la JPK Figure 4.9). Each reachable region should be stated in terms of its list representation.

[4] see JPH definition 52