

rtl bugs report

Celso
Christophe

15 septembre 2003

1 Génération du DTA

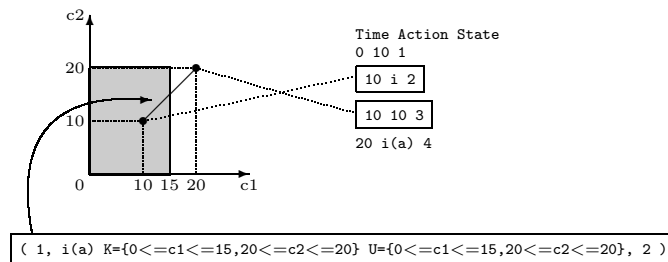
Les clocks associées à `delay` sont construites comme celles de `latency` : elles recouvrent la limitation de l'offre temporelle !

<pre>hide a in ((delay(10) a{15}; stop) [a] (delay(10) i; a; stop)) [a] (delay(20); a{0}; stop)</pre>	<pre>Time Action State 0 10 1 10 i 2 10 10 3 20 i(a) 4</pre>	<pre>State 0 : 3 clocks State 1 : 2 clocks State 2 : 0 clocks 3 states, 2 arcs (0, i K={10<=c2<=10} U={10<=c2<=10} theta={{(1,1),(3,2)}, 1 } (1, i(a) K={10<=c1<=25,20<=c2<=20} U={10<=c1<=25,20<=c2<=20}, 2)</pre>
<pre>hide a in (delay(10) (a{15}; stop) [a] (i; a; stop)) [a] (delay(20); a{0}; stop)</pre>	<pre>Time Action State 0 10 1 10 i 2 10 10 3 20 i(a) 4</pre>	<pre>State 0 : 3 clocks State 1 : 2 clocks State 2 : 0 clocks 3 states, 2 arcs (0, i K={10<=c2<=10} U={10<=c2<=10} theta={{(1,1),(3,2)}, 1 } (1, i(a) K={0<=c1<=15,20<=c2<=20} U={0<=c1<=15,20<=c2<=20}, 2)</pre>

Les deux spécifications RT-LOTOS sont équivalentes (distributivité de `delay`) ; les traces de simulations aussi ; pourtant les DTA diffèrent !

Le deuxième DTA est faux !

Lorsque l'on arrive dans l'état 1, toutes les clocks sont à 10. Le temps va encore progresser de 10 alors que l'on a $0 \leq c1 \leq 15$!!!



- ⇒ soit un bug de la fonction `theta`
 - le DTA serait juste si il n’y avait pas `theta={ (1,1), ... }` à l’état 0
- ⇒ soit un bug sur la construction des domaines
 - le DTA serait juste si les bornes de `c1` à l’état 1 étaient incrémentées de `c1` à l’état 0

2 Simulation d’actions observables

Une action `a{t}` $t \in [0, \infty]$, est offerte pour une période limitée t . Si pour une raison quelconque, l’action `a` ne peut se produire pendant cet intervalle de temps, il y a violation temporelle.

Il semble cependant que le simulateur `rtl` (même avec `-SIM-1`) tire toujours les actions observables dans `KUU`!

Exemple : `latency(t1) a{t2}; stop`

$t_1 \leq t_2$: DTA = (0, a $K=\{0 \leq c1 \leq t_2\}$ $U=\{t_1 \leq c1 \leq t_2\}$, 1) $\Rightarrow K \subseteq U$
L’action `a` est *toujours* tirée entre 0 et t_2 !!! Normalement, le temps devrait pouvoir avancer au-delà de t_2 sans que l’on tire `a`...

$t_1 > t_2$: DTA = (0, a $K=\{0 \leq c1 \leq t_2\}$ $U=\{\text{Inf} < c1\}$, 1) $\Rightarrow K \cap U = \emptyset$
Le temps peut parfois avancer au-delà de t_2 (au maximum de 1! pour-quoi?) avant de tirer `a` → situation normale (i.e. violation temporelle).
Est-ce parce-que U commence à ∞ ?

3 Simulation de `a` \equiv `a{∞}`

Si l’action `a` est observable, le processus `a; stop` signifie `a{∞}; stop`. Pourtant `rtl`, même avec l’option `-SIM-1` s’obstine à tirer l’action `a` à l’instant 0!!!.

Rappel concernant `-SIM-1` Par défaut, le simulateur `rtl` force l’environnement à toujours offrir immédiatement les actions observables, c’est à dire qu’elles se comportent comme si elles étaient internes (ce qui est le plus pratique...).

Au contraire, avec l’option `-SIM-1`, l’environnement offre les actions observables quand bon lui semble.

4 Génération du DTA avec des hide dans les processus

La génération du DTA boucle sur une spec du style :

```
hide a,b in
  P[a,b]

where
  process P[a,b] : noexit :=
    hide c,d in
      a;c;
      b;d;
      P[a,b]
  endproc
```

et pas sur :

```
hide a,b,c,d in
  P[a,b,c,d]

where
  process P[a,b,c,d] : noexit :=
    a;c;
    b;d;
    P[a,b,c,d]
  endproc
```