Le temps réel dans AltaRica: Timed AltaRica

Franck Cassez*, Claire Pagetti** et Olivier Roux*

* {name}@irccyn.ec-nantes.fr

** pagetti@labri.fr

IRCCyN - Ecole Centrale de Nantes

Le langage AltaRica

- langage de description de haut niveau
- synchronisations *à la Arnold-Nivat*, *broadcast*
- hiérarchie
- variables partagées
- priorité

Le langage AltaRica

- langage de description de haut niveau
- synchronisations *à la Arnold-Nivat*, *broadcast*
- hiérarchie
- variables partagées
- priorité

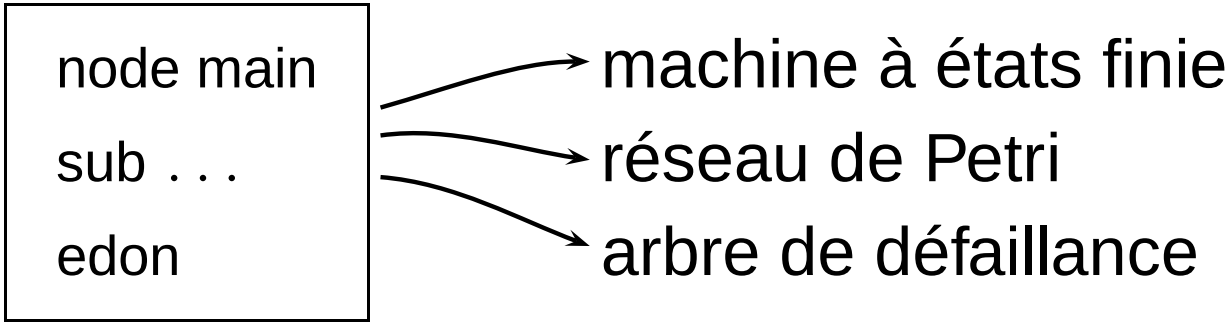
modèle AltaRica

```
node main
sub . . .
edon
```

Le langage AltaRica

- langage de description de haut niveau
- synchronisations *à la Arnold-Nivat*, *broadcast*
- hiérarchie
- variables partagées
- priorité

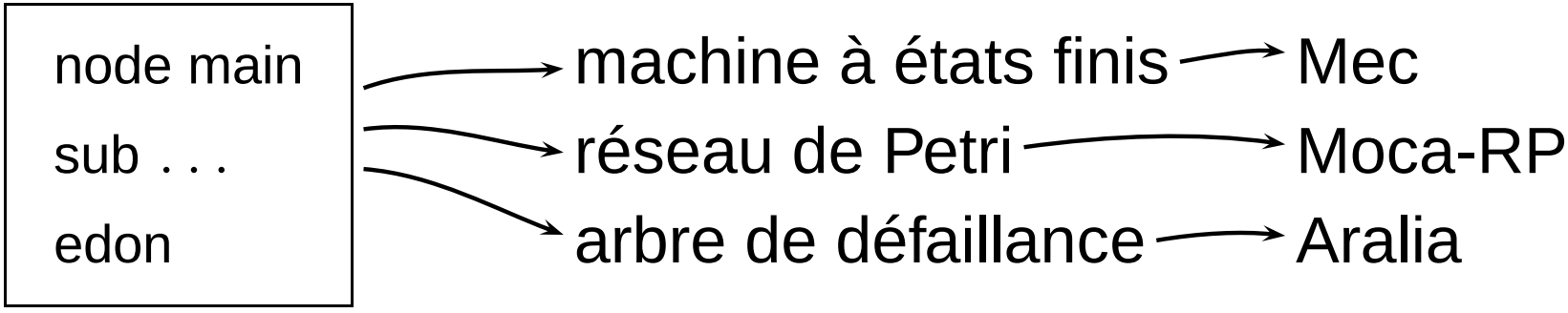
modèle AltaRica compilation



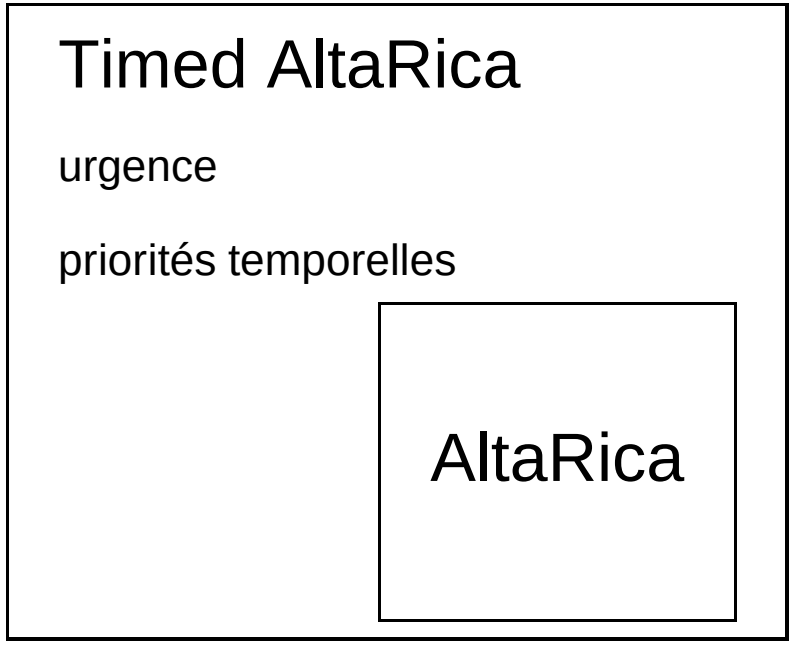
Le langage AltaRica

- langage de description de haut niveau
- synchronisations *à la Arnold-Nivat*, *broadcast*
- hiérarchie
- variables partagées
- priorité

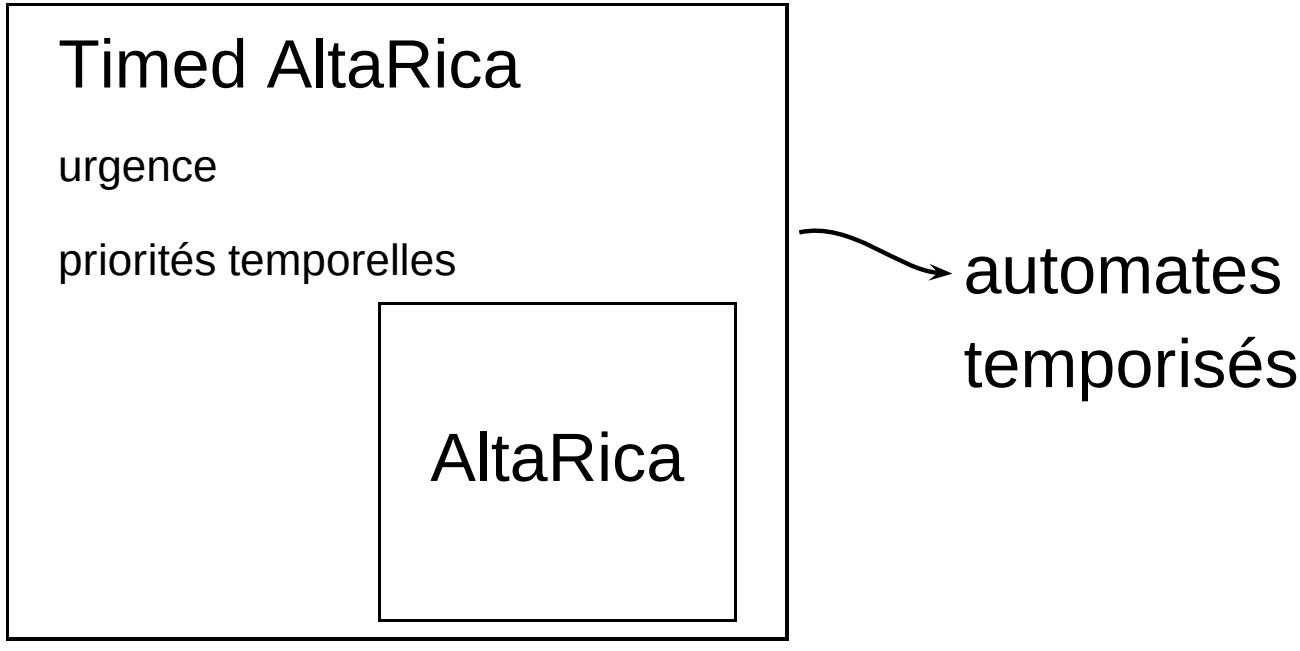
modèle AltaRica compilation outils



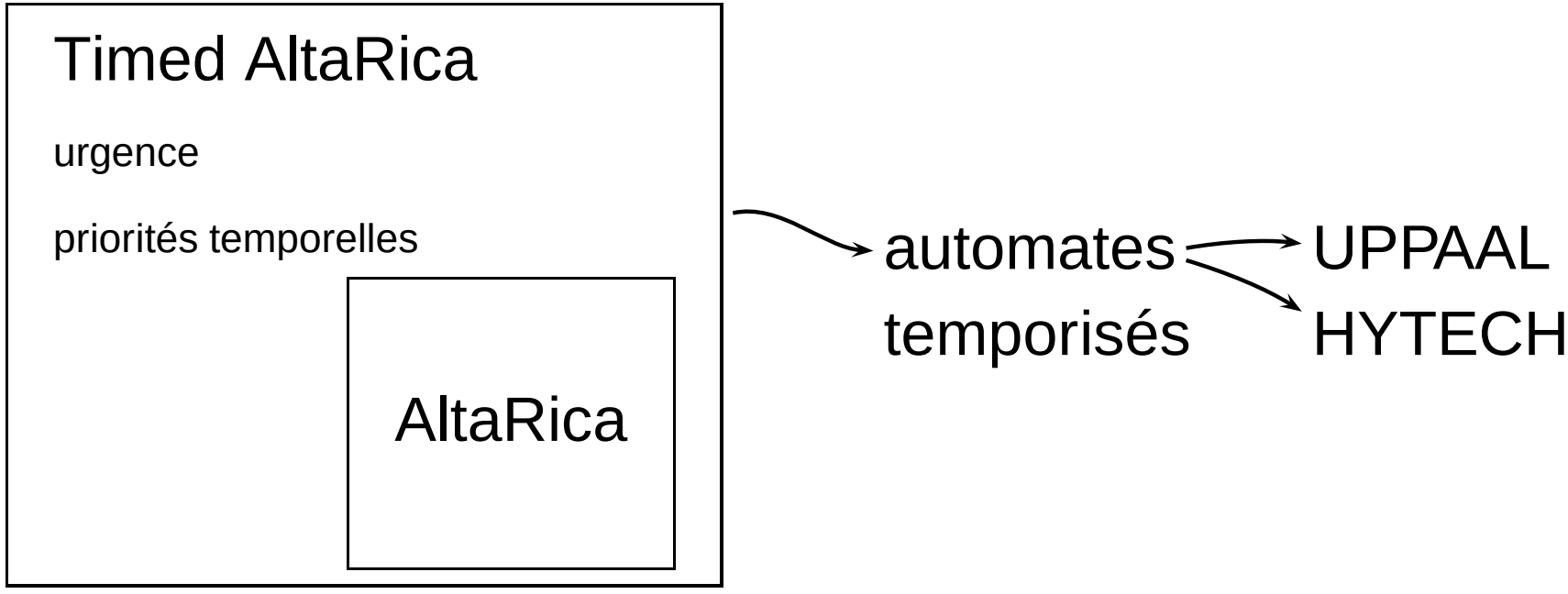
Timed AltaRica



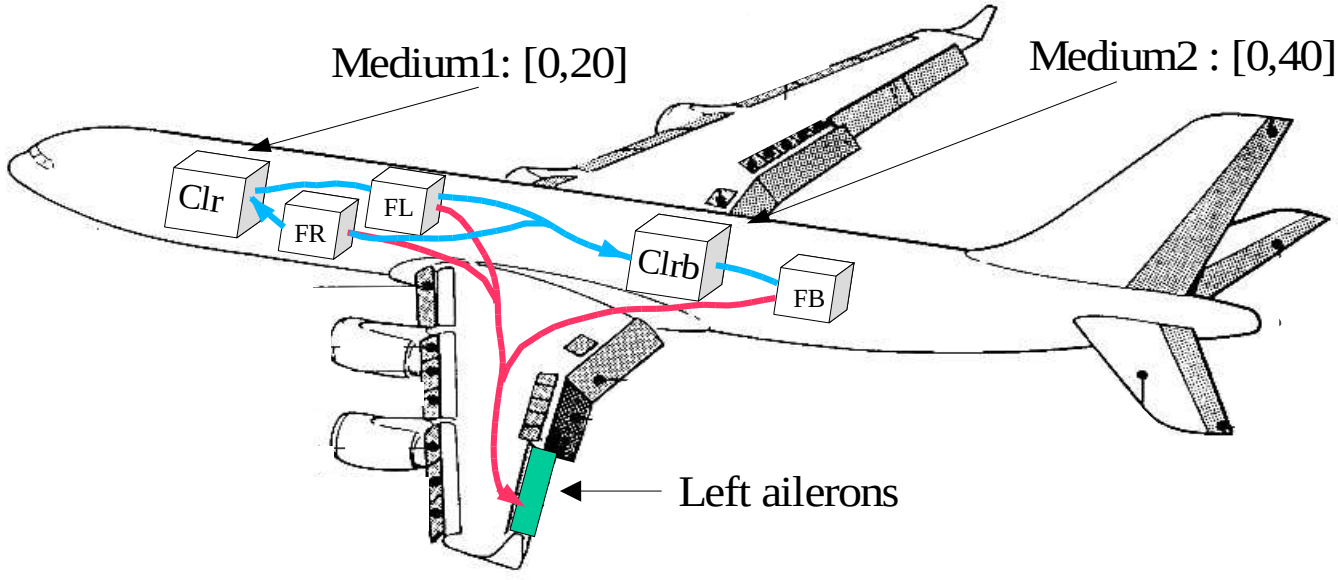
Timed AltaRica



Timed AltaRica



Exemple [BE]



Un composant (syntaxe)

node CLR

edon

Un composant (syntaxe)

`node` CLR

variable discrète — horloge

déclaration des variables

`state` $panne : [0,1]$; $h : clock$;

`flow` $Panne_R, Panne_{LR} : [0,1]$

`edon`

Un composant (syntaxe)

node CLR
state $panne : [0,1]; \ h : clock;$
flow $Panne_R \ , \ Panne_{LR} : [0,1];$

déclaration des événements { event $refresh$

edon

Un composant (syntaxe)

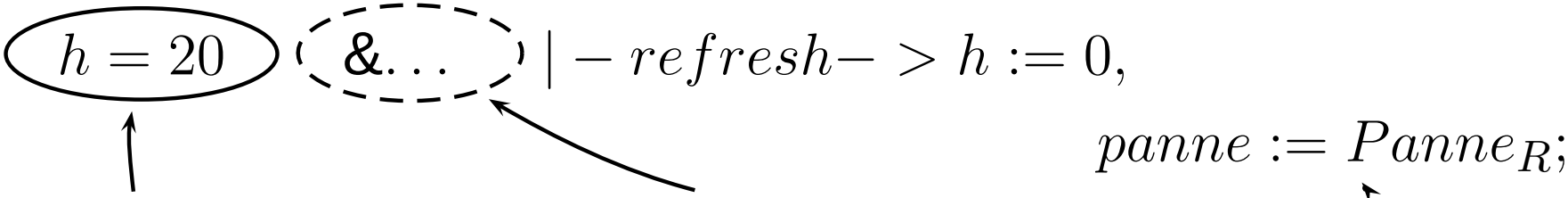
`node` CLR

`state` $panne : [0,1]; \ h : clock;$

`flow` $Panne_R \ , Panne_{RL} : [0,1];$

`event` $refresh$

déclaration des transitions $\left\{ \right.$ `trans`



`edon`

Un composant (syntaxe)

node CLR

state $panne : [0,1]; \ h : clock;$

flow $Panne_R \ , \ Panne_{RL} : [0,1];$

event $refresh$

trans $h = 20 \mid - refresh - > \ h := 0, \ panne := Panne_R;$

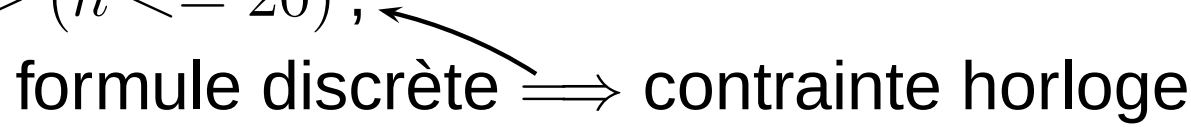
contraintes
- initiale
  - `init` $panne := 0 \ , \ h := 0$
- globales
  - `assert` $Panne_{LR} = panne$
  - `tinvariant` $true => (h <= 20)$

edon

Un composant (syntaxe)

node CLR
state $panne : [0,1];\ h : clock;$
flow $Panne_R\ ,\ Panne_{RL} : [0,1];$
event $refresh$
trans $h = 20 \mid -refresh-> \ h := 0,\ panne := Panne_R;$
init $panne := 0\ ,\ h := 0$
assert $Panne_{LR} = panne$

tinvariant $true => (h <= 20)\ ;$



edon

Un composant (sémantique)

valuation = assigne une valeur à chaque variable

exemple :

$Panne_R$	$\rightarrow$	0	0	0	0
$Panne_{LR}$	$\rightarrow$	0	0	1	0
$panne$	$\rightarrow$	1	0	1	0
h	$\rightarrow$	10	30	10	10
		non admissible		non atteignable	admissible

valuation admissible $\longleftrightarrow$ configuration

Un composant (sémantique)

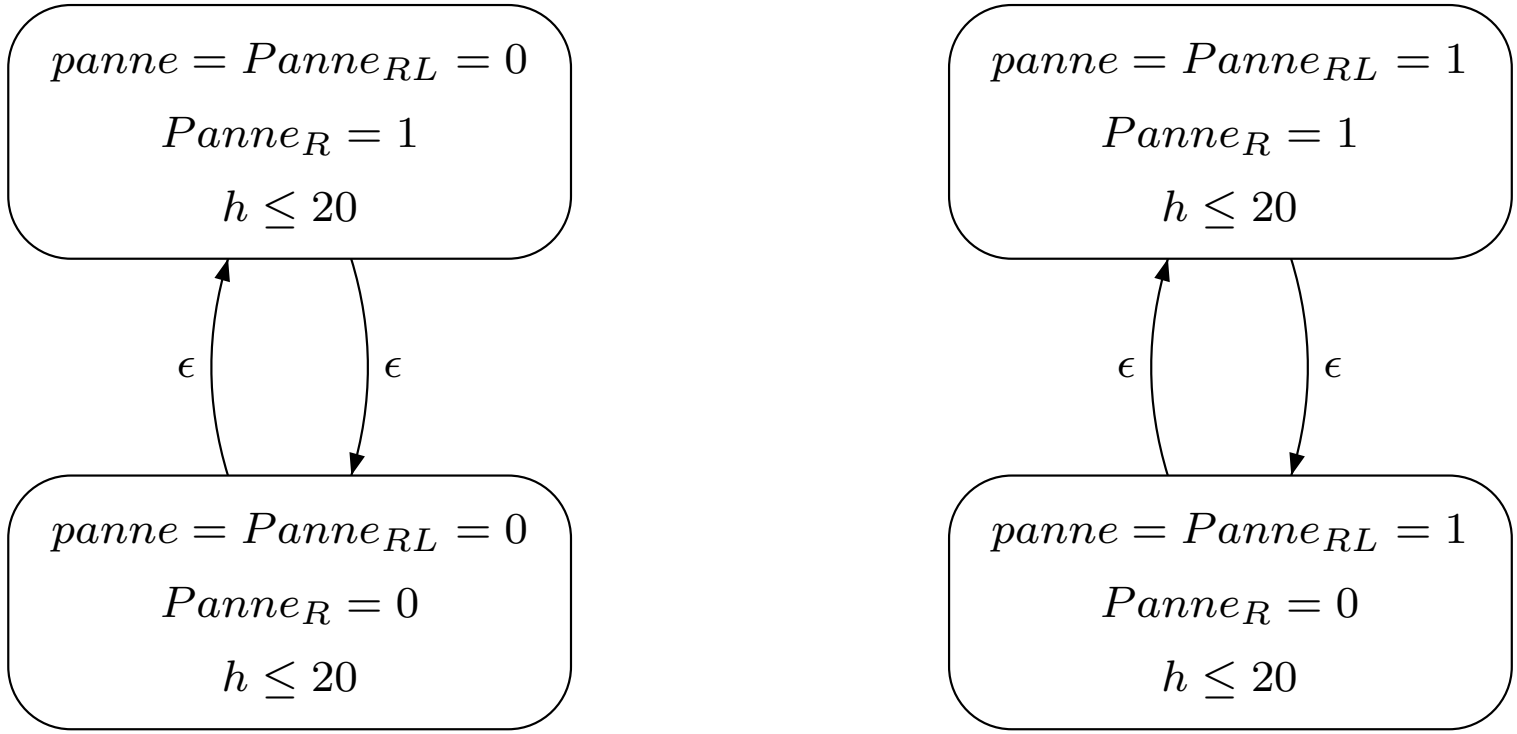
$$panne = Panne_{RL} = 0$$
$$Panne_R = 1$$
$$h \leq 20$$

$$panne = Panne_{RL} = 1$$
$$Panne_R = 1$$
$$h \leq 20$$

$$panne = Panne_{RL} = 0$$
$$Panne_R = 0$$
$$h \leq 20$$

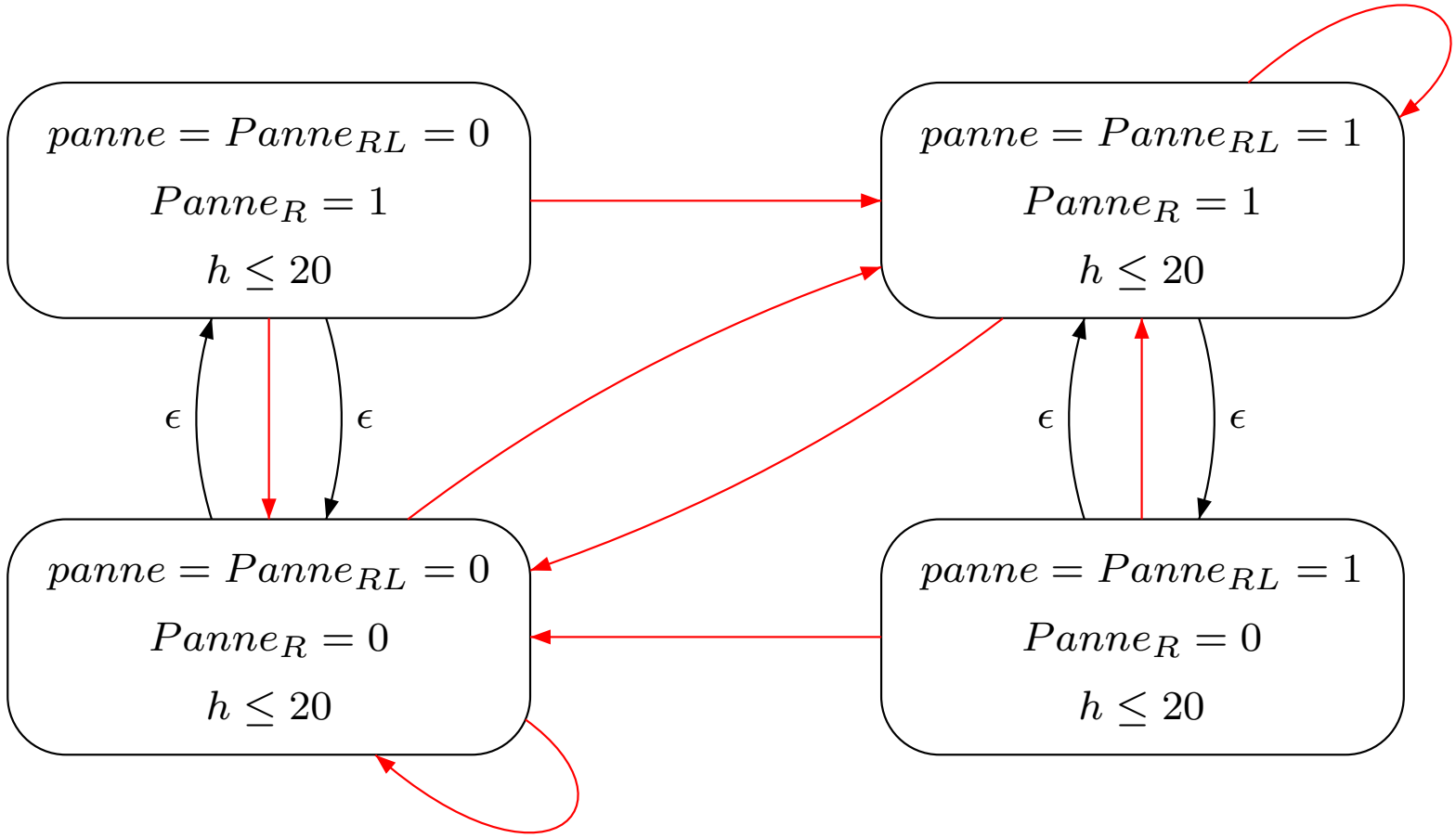
$$panne = Panne_{RL} = 1$$
$$Panne_R = 0$$
$$h \leq 20$$

Indéterminisme dû aux flux



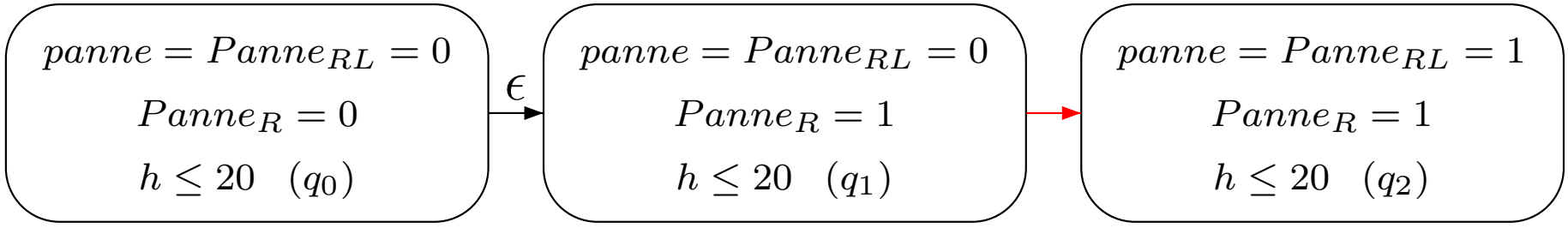
Sémantique du composant

- $h = 20, refresh, h := 0$



Un composant (sémantique)

Modèle des automates temporisés [AD90,AD94]

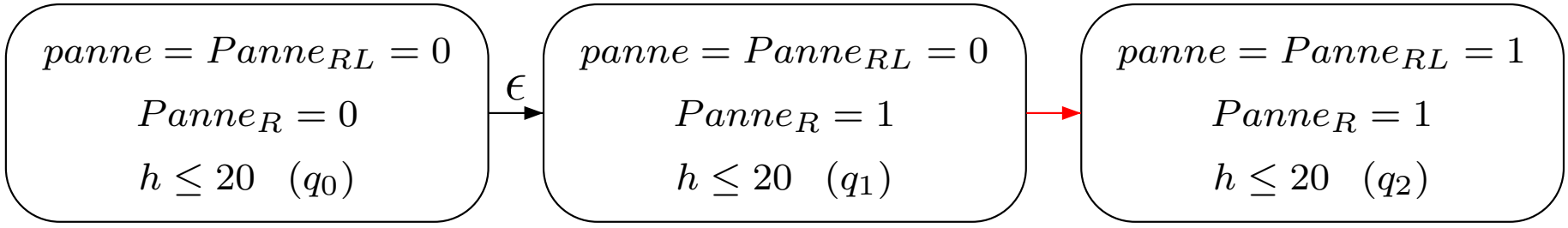


– $h = 20, refresh, h := 0$

Une exécution :

Un composant (sémantique)

Modèle des automates temporisés [AD90,AD94]



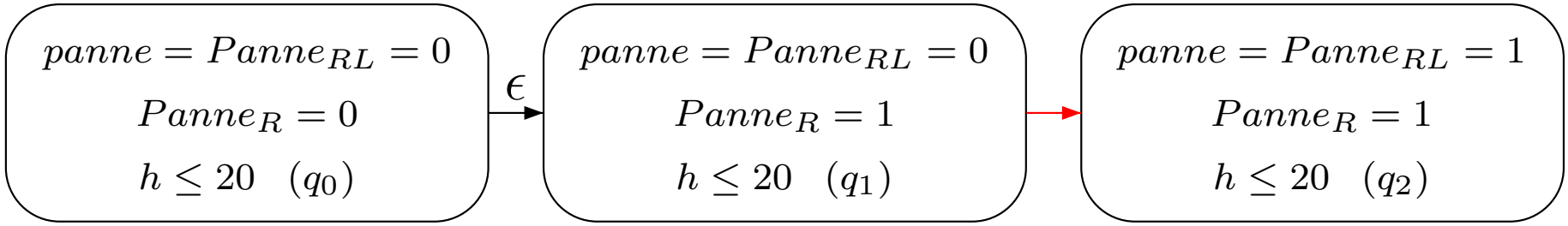
– $h = 20, refresh, h := 0$

Une exécution :

	q_0
h	0

Un composant (sémantique)

Modèle des automates temporisés [AD90,AD94]



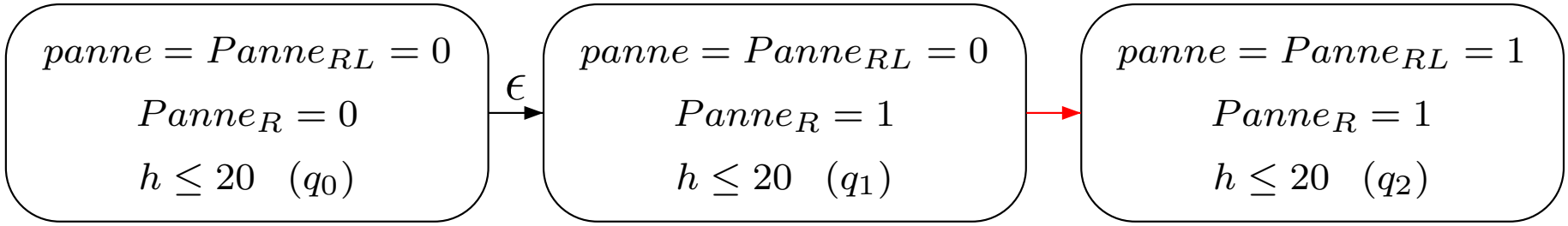
– $h = 20, refresh, h := 0$

Une exécution :

$$\begin{array}{lll} & q_0 & \xrightarrow{\delta(10.1)} \quad q_0 \\ \text{h} & 0 & \qquad\qquad 10.1 \end{array}$$

Un composant (sémantique)

Modèle des automates temporisés [AD90,AD94]



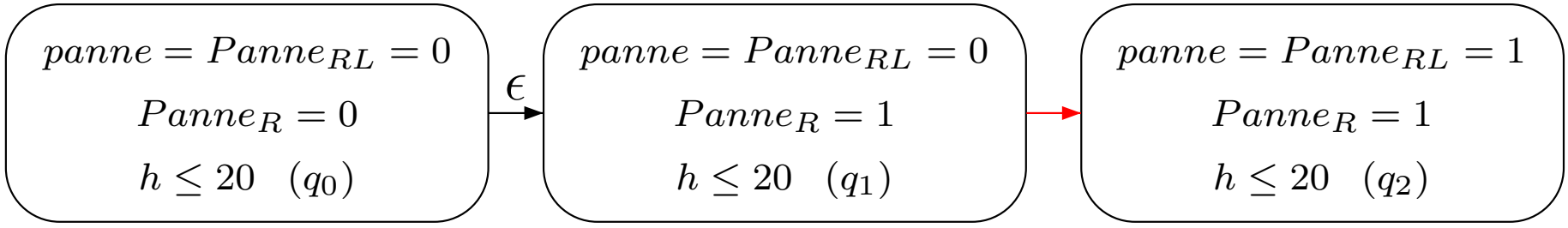
– $h = 20, refresh, h := 0$

Une exécution :

$$\begin{array}{lccccc} & q_0 & \xrightarrow{\delta(10.1)} & q_0 & \xrightarrow{\epsilon} & q_1 \\ \text{h} & 0 & & 10.1 & & 10.1 \end{array}$$

Un composant (sémantique)

Modèle des automates temporisés [AD90,AD94]



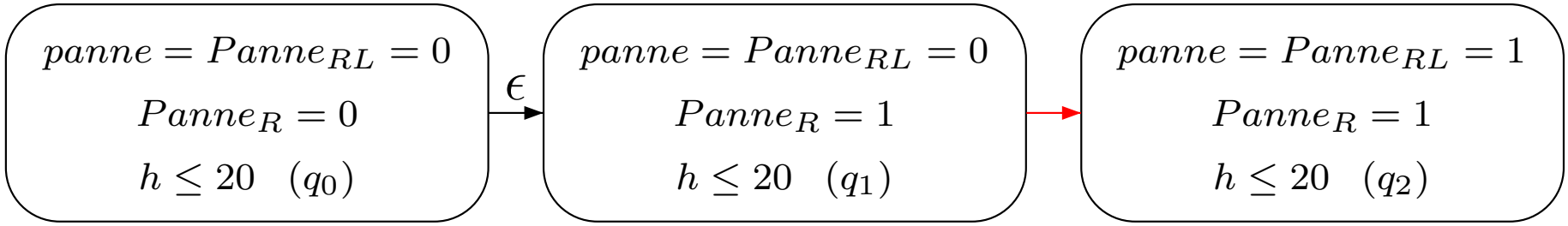
– $h = 20, refresh, h := 0$

Une exécution :

$$\begin{array}{cccccccc} & q_0 & \xrightarrow{\delta(10.1)} & q_0 & \xrightarrow{\epsilon} & q_1 & \xrightarrow{\delta(2.3)} & q_1 \\ \text{h} & 0 & & 10.1 & & 10.1 & & 12.4 \end{array}$$

Un composant (sémantique)

Modèle des automates temporisés [AD90,AD94]



– $h = 20, refresh, h := 0$

Une exécution :

		$\delta(10.1)$		ϵ		$\delta(2.3)$		refresh	
	q_0	$\longrightarrow$	q_0	$\longrightarrow$	q_1	$\longrightarrow$	q_1	$\longrightarrow$	q_2 $\quad \cdots$
h	0		10.1		10.1		12.4		0

Système de transitions temporisé interfacé

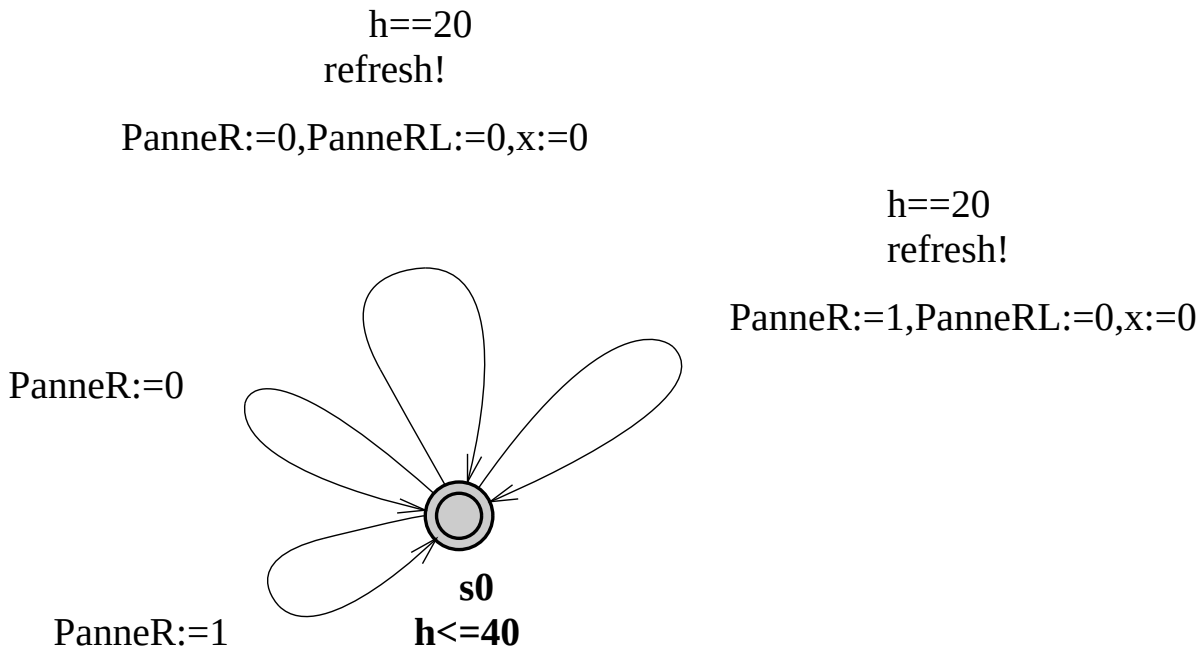
$\mathcal{A} = \langle E_t, F_t, S_t, \pi, T \rangle$

1. $E_t = E_+ \cup \{\epsilon\} \cup \mathbb{R}_{\geq 0}$,
2. $F_t = F \times \mathbb{R}^m_{\geq 0}$ valeurs de flux et $S_t = S \times \mathbb{R}^n_{\geq 0}$ états ;
3. $\pi : S_t \rightarrow 2^{F_t}$ avec $\pi(q)$ toutes les valeurs de flux admissibles en q. On suppose que $\forall q \in S_t,\ \pi(q) \neq \emptyset$.
4. $T \subseteq S_t \times F_t \times E_t \times S_t$
   (a) $(q, g, e, q') \in T \Rightarrow g \in \pi(q)$
   (b) $\forall q \in S_t, \forall g \in \pi(q),\ (q, g, \epsilon, q) \in T$
   (c) $\forall q \in S_t, \forall g \in \pi(q),\ (q, g, 0, q) \in T$

Configuration $(q, g) \in S_t \times F_t$ avec $g \in \pi(q)$.

Calcul symbolique

On regroupe les valuations satisfaisant le même invariant temporel.



Urgence

node FL

flow $Panne_{RL}, Panne_L : [0,1]$

event $breakdown_L, action_L, Cmd$

priority $action_L > time$

state $etat, panne : [0,1]; y : clock$

trans $etat = 0 \mid - breakdown_L - > \ panne := 1;$

$Panne_{RL} = 1 \ \& \ panne = 0 \ \& \ etat = 0 \mid - action_L - > \ etat := 1, y := 20;$

$etat = 1 \mid - breakdown_L - > panne := 1, etat := 0;$

$etat = 1 \ \& \ y = 20 \mid - Cmd - > y := 0$

init $etat := 0, panne := 0$

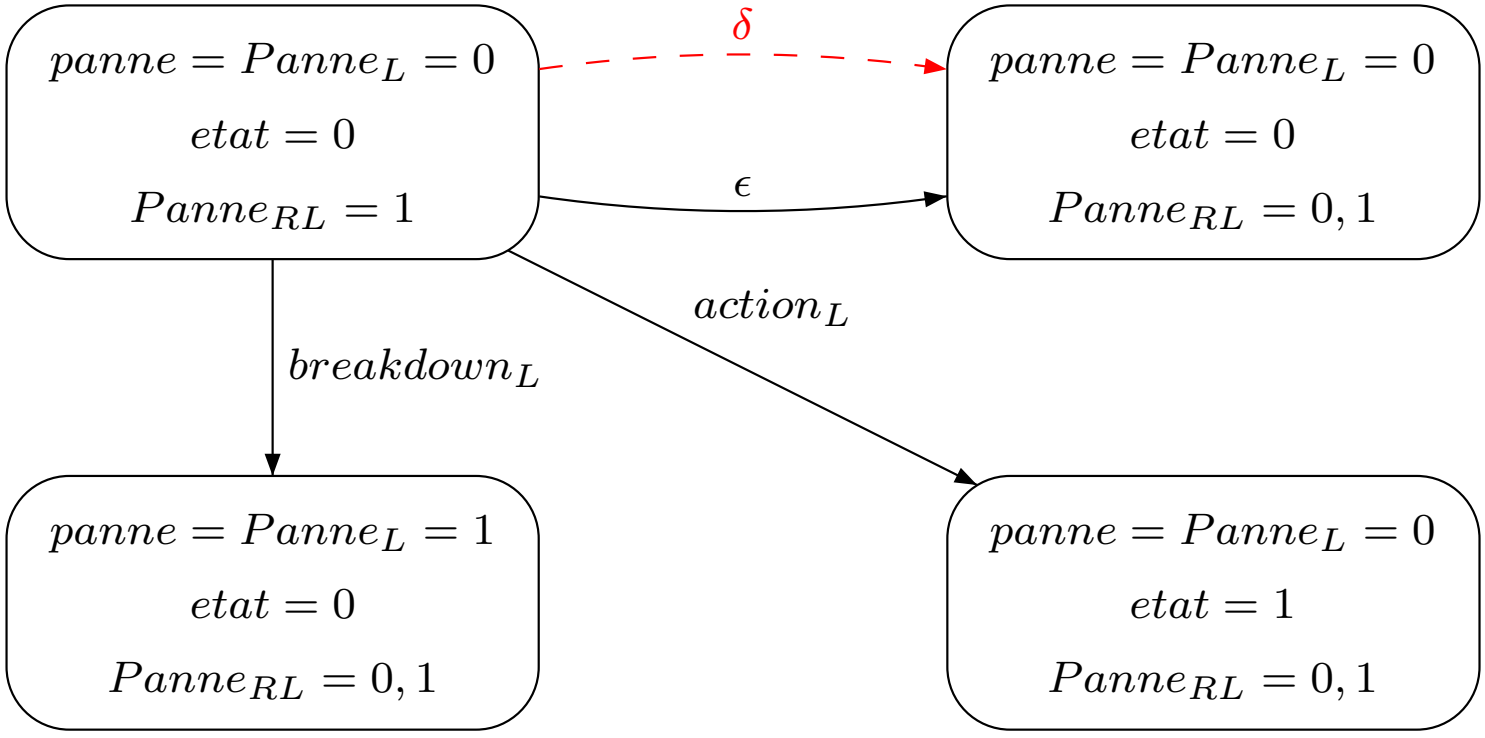
assert $Panne_L = panne;$

tinvariant $etat = 1 \ \& \ panne = 0 => y <= 20$

edon

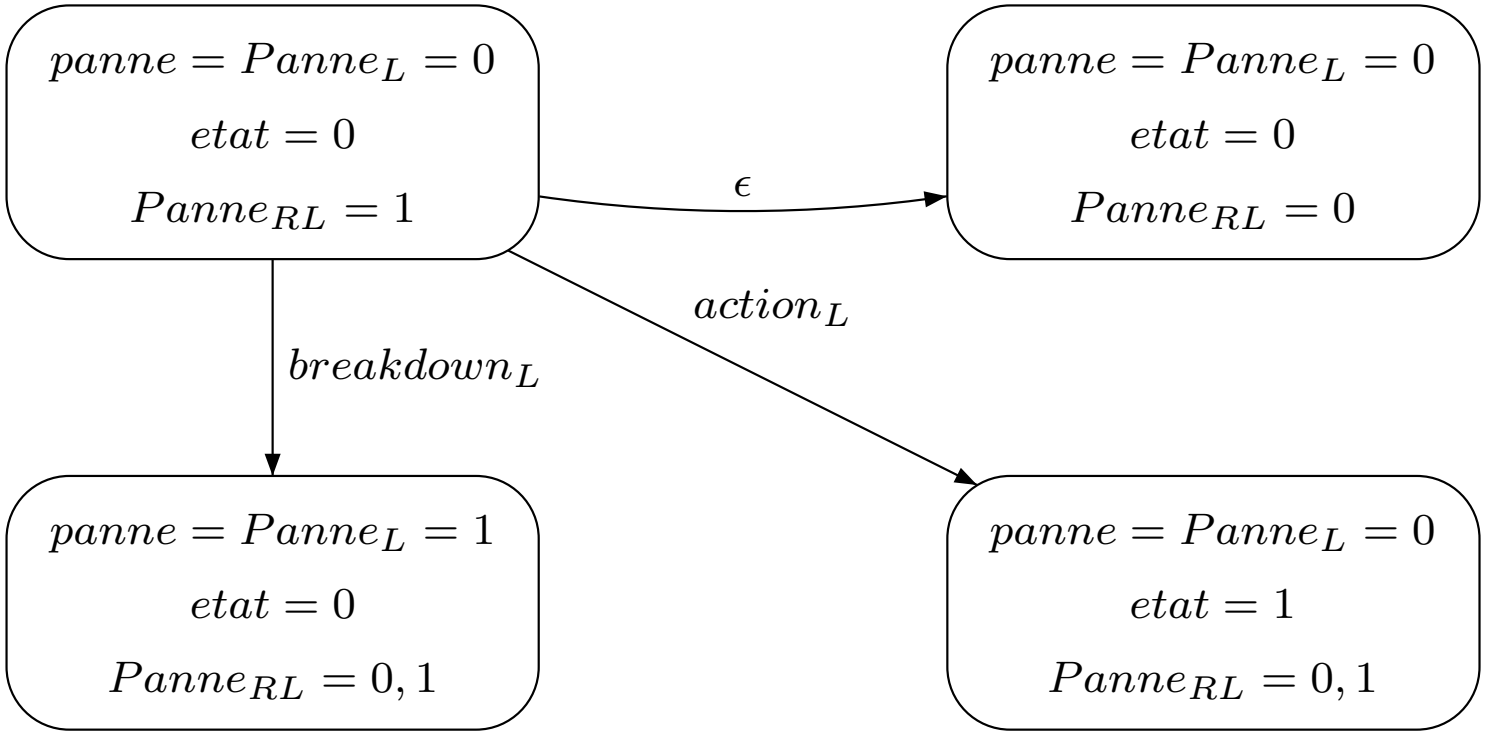
Urgence

Le composant FL sans urgence



Urgence

Le composant FL avec urgence

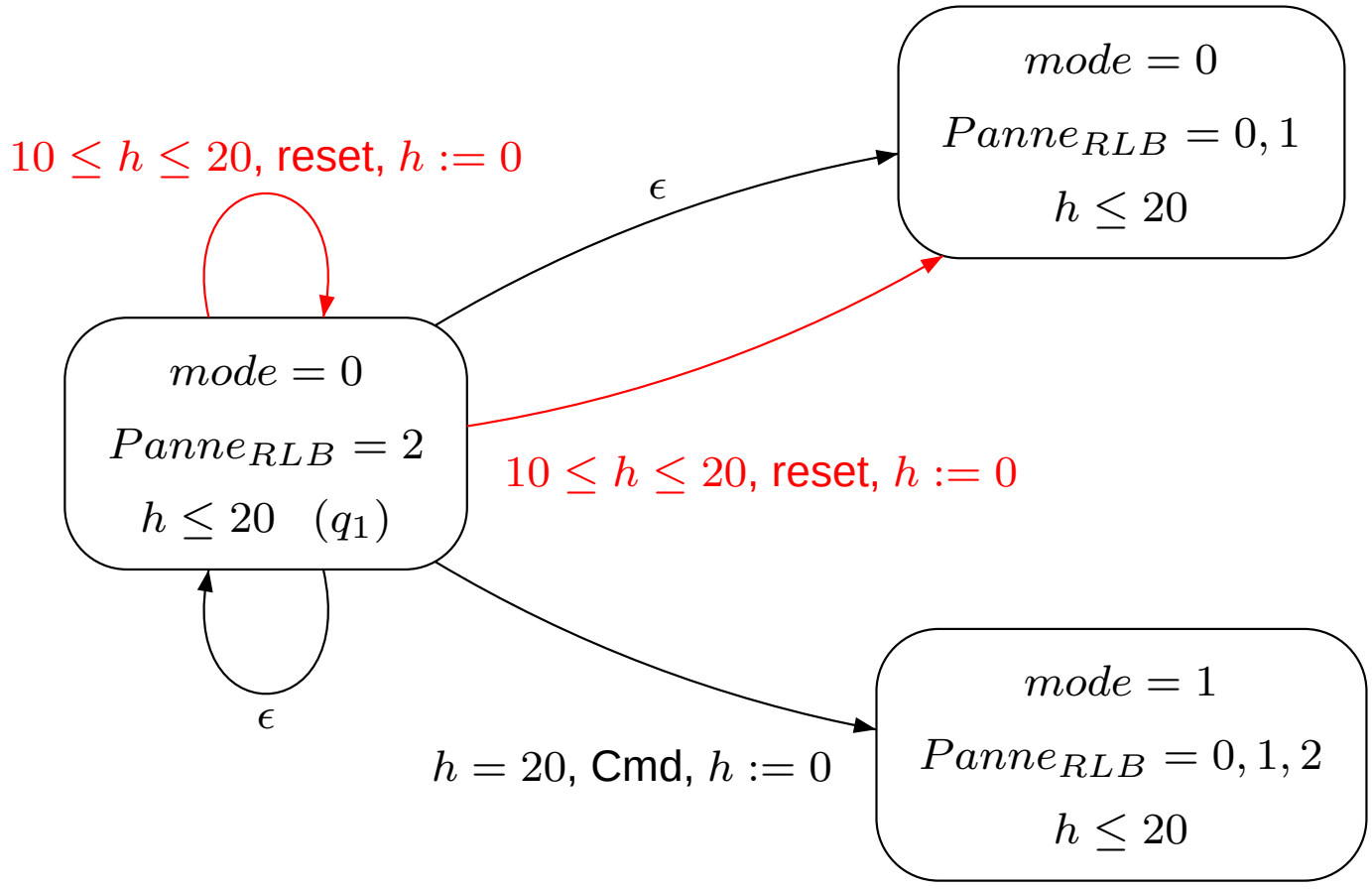


Priorité temporelle

```
node     FB
flow     Panne_RLB : [0, 2];
state    mode : [0, 1]; h : clock;
event    reset, Cmd
priority      reset (<, 5) Cmd
trans    mode = 0 & 10 <= h <= 20 | − reset− > h := 0;
         mode = 0 & Panne_RLB = 2&h = 20 | − Cmd− > h := 0, mode := 1;
         mode = 1 & h = 20 | − Cmd− > h := 0;
init     mode := 0, h := 0
tinvariant    true => (h <= 20)
edon
```

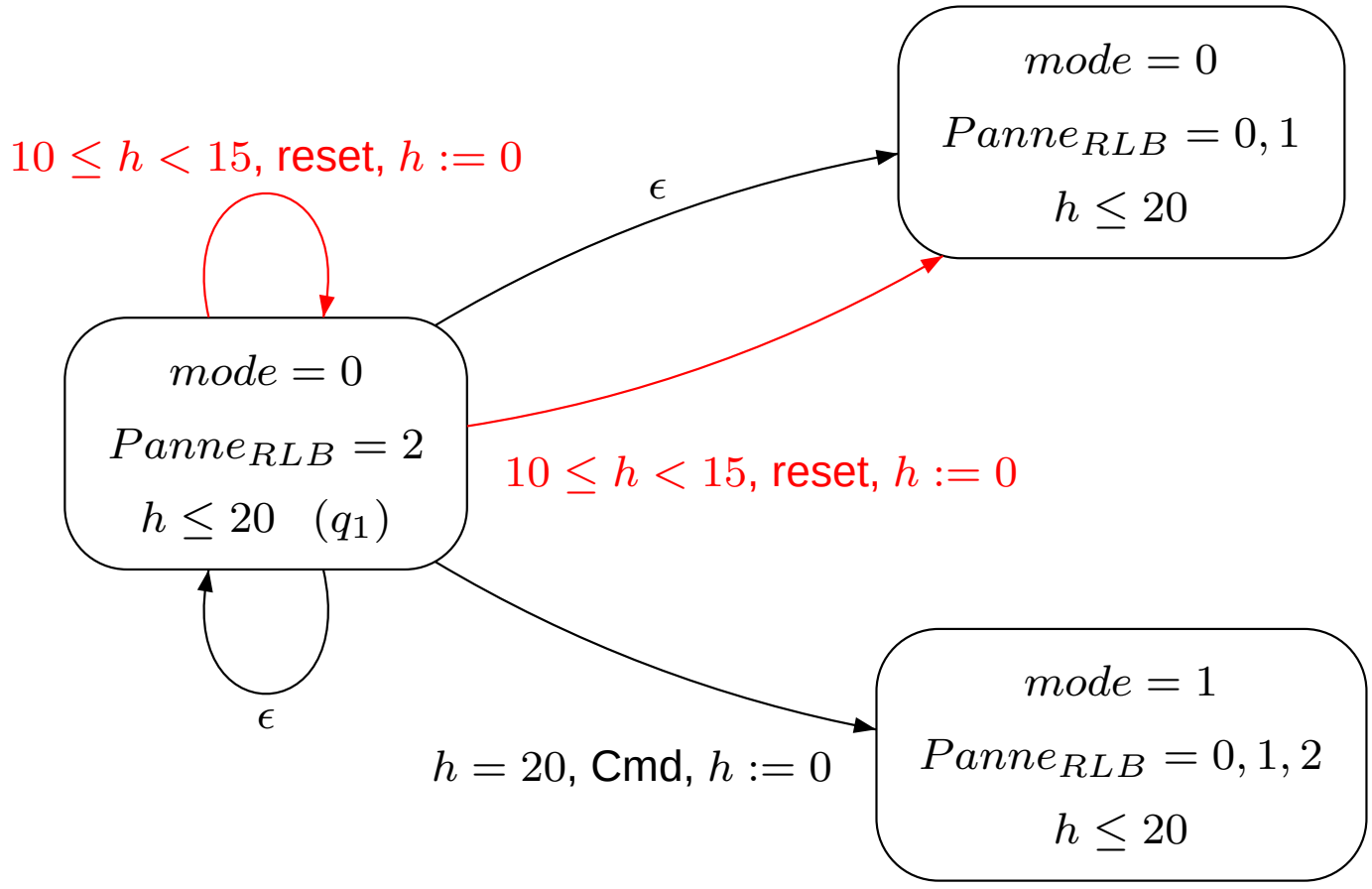
Priorité temporelle

F_B sans priorité temporelle

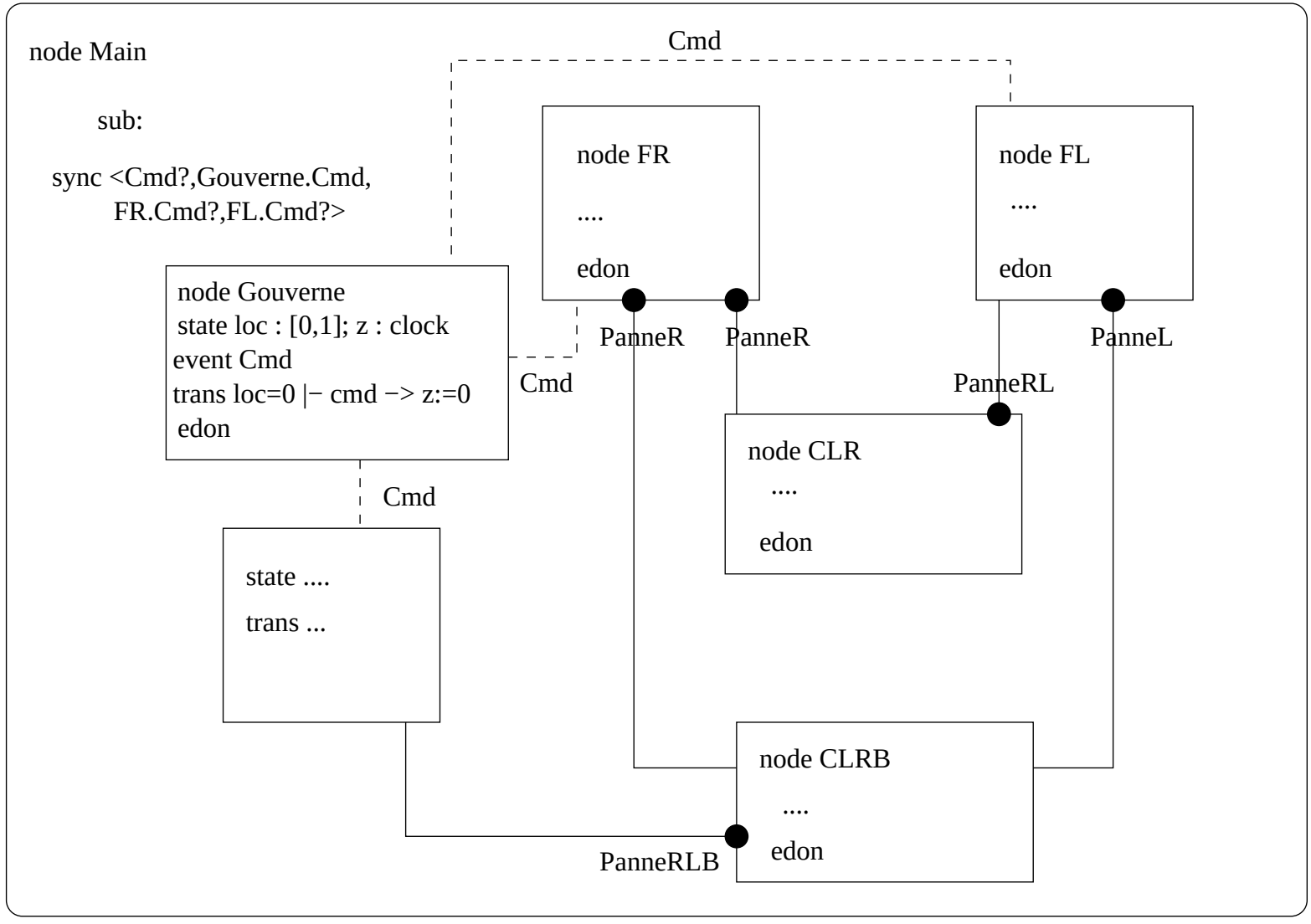


Priorité temporelle

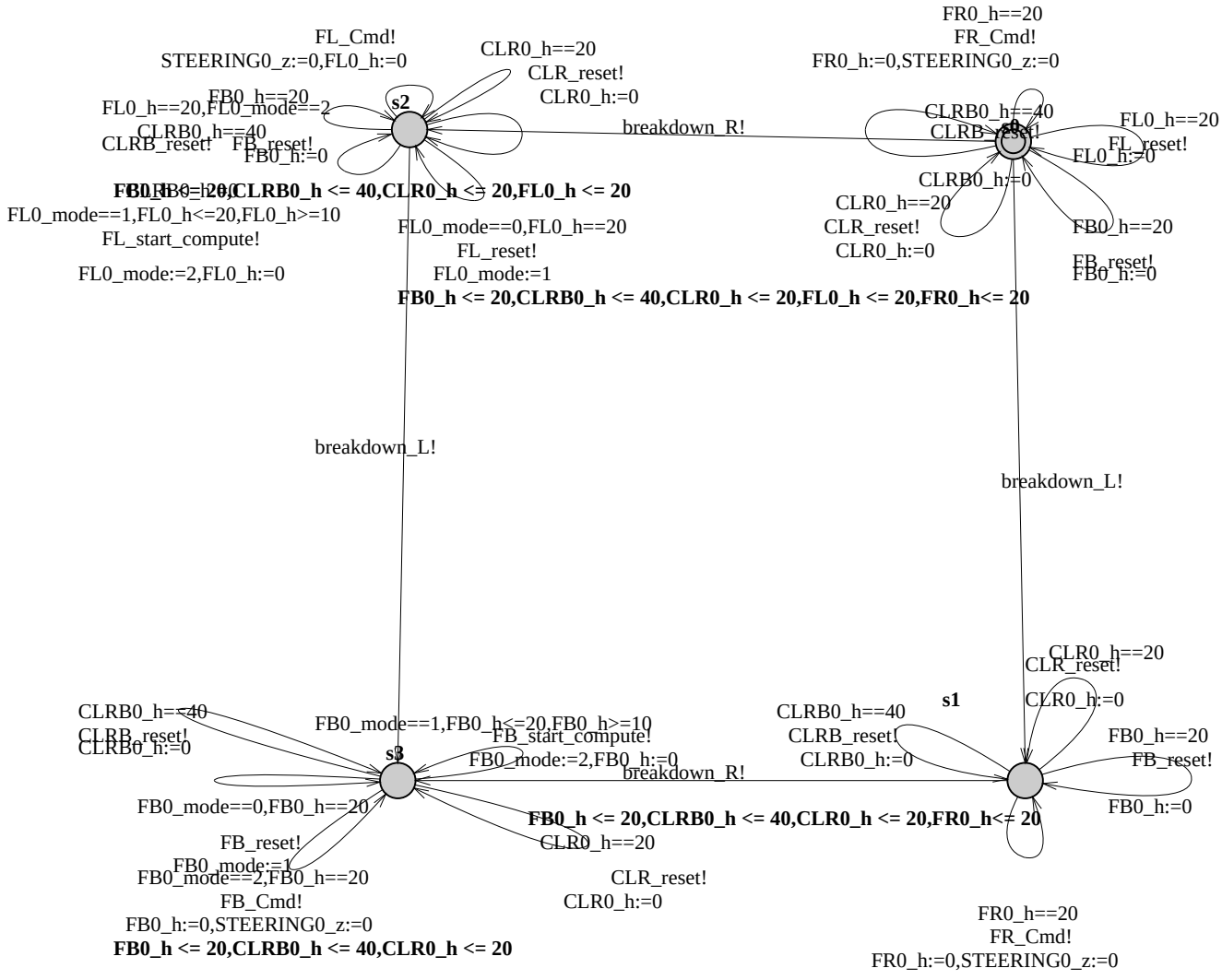
F_B avec priorité temporelle



Hiérarchie

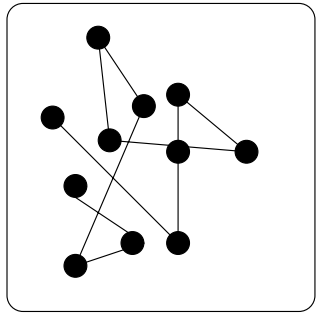

node Main
sub:
sync <Cmd?,Gouverne.Cmd,
FR.Cmd?,FL.Cmd?>
Cmd
node FR
....
edon
node FL
....
edon
node Gouverne
state loc : [0,1]; z : clock
event Cmd
trans loc=0 |− cmd −> z:=0
edon
Cmd
PanneR
PanneR
PanneL
PanneRL
node CLR
....
edon
Cmd
state ....
trans ...
node CLRB
....
edon
PanneRLB

Hiérarchie - mise à plat



Vérification

Model Checking



Système $\models$ satisfait-il ϕ la propriété ?

Méthode formelle : analyse exhaustive des comportements du système.

Intérêt : éviter des mauvais comportements (crash) dans les systèmes *critiques* et *réactifs*.

Vérification

Logique temporelle *temporisée* pour exprimer propriétés :
TCTL

Outil : UPPAAL

Vérification

Logique temporelle *temporisée* pour exprimer propriétés :
TCTL

Outil : UPPAAL

Pas de blocage : $AG(not\ deadlock)$

Vérification

Logique temporelle *temporisée* pour exprimer propriétés :
TCTL

Outil : UPPAAL

P1 : à tout instant, il y a au plus une fonction en mode commande

$$AG\big((FR.etat = 0 \implies (FL.etat = 0 \wedge etat = 0)) \wedge ((FL.etat = 1 \wedge FL.panne = 0) \implies etat = 0)\big) ;$$

Vérification

Logique temporelle *temporisée* pour exprimer propriétés :
TCTL

Outil : UPPAAL

P2 : la gouverne ne doit pas rester non commandée plus de 160 ms.
$AG(Gouverne.z \leq 60)$

Conclusion

Un langage Timed AltaRica de modélisation hiérarchique pour les systèmes réactifs, critiques, temps réel et distribués.
Expressivité = celle des automates temporisés.

Un prototype TARC met à plat et traduit des spécifications Timed AltaRica en automates temporisés.
Vérification formelle.

A venir

- implantation priorités temporelles
- extension hybride

Bibliographie

[BE] F. Boniol et J. Ermont, « Modelling and Verifying Embedded Systems Sensitive to Resource Availability », 3rd European Systems Engineering Conference, EuSEC'2002, Toulouse, France.

[AD90] R. Alur et D. Dill, « Automata for modelling real-time systems », 17 International Conference on Automata, Languages, and Programming, volume 443, pages 322–335, Lecture Notes in Computer Science, SpringerVerlag, 1990.

[AD94] R. Alur, et D. Dill,« A theory of timed automata », Theoretical Computer Science B, volume 126, pages 183–235, 1994.