Une Extension Temporelle d'AltaRica

Claire Pagetti
Sous la direction d'O. Roux et F. Cassez.
IRCCyN - Nantes

Plan

1. Systèmes temporisés
2. Exemple
3. AltaRica temporisé
4. Vérification temporelle

De l'intérêt du temps réel

Système temps réel : interaction des composants entre eux et avec un environnement non contrôlable.

2 contraintes temps réel caractéristiques

réactivité réponse à des stimuli extérieurs

ponctualité respect des échéances

$\Longrightarrow$ émergence de variables réelles

Systèmes temporisés

Système temporisé = système de transitions particulier admettant des actions discrètes et continues (correspondant à un écoulement du temps).

Dans la pratique, système de transitions + horloges.
Modèle des automates temporisés Alur et Dill [**?**].

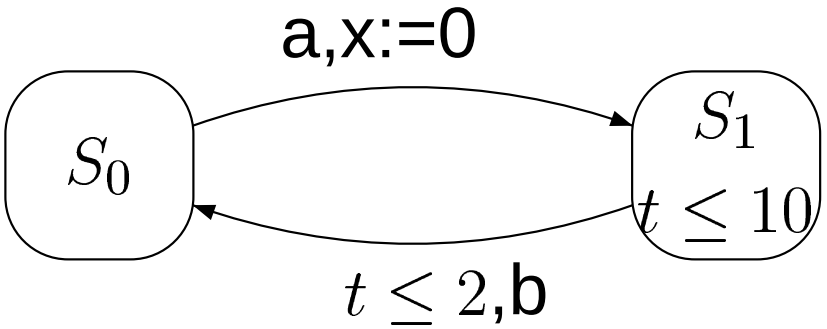
Horloge : variable à valeurs réelles, augmente continûment avec le temps, toutes les horloges croissent à la même vitesse et les seules opérations possible sont le test et le reset lors d'une transition.

Systèmes temporisés

Contraintes d'horloge : $g := x \backsim r | x - y \backsim r | g \wedge g | g \vee g$ *avec* $\backsim \in \{<, <, >, \geq, =\},\ r \in \mathbb{Q}$

Décidabilité : accessibilité avec automates temporisés avec update

Intérêt d'AltaRica temporisé : remplir le manque d'un langage de haut niveau compilant vers des automates temporisés



Priorités temporelles

Etendre les priorités statiques d'AltaRica en introduisant des priorités prenant en compte des informations temporelles.
3 types standards de transitions (Sifakis) :

eagerness : si la transition est tirable, alors le temps ne peut plus s'écouler et la transition doit être tirer immédiatement

delayable : la transition peut être tirée tant que la garde est vraie et avant une deadline

lazy : transition sans deadline (éventuellement jamais)

Plan

1. Systèmes temporisés
2. Exemple
3. AltaRica temporisé
4. Vérification temporelle

Exemple

```
node TRAIN
 flow   N [0,1];
 event  approach, in, exit;
 state  etat [0,2], n [0,1], t clock;
 trans  t>=70 & etat=0 |- approach ->
            etat := 1, t := 0, n := 1;
        20<=t<=30 & etat=1  |- in ->
             etat := 2, t := 0;
        10<=t<= 20 & etat=2  |- exit ->
             etat := 0, t := 0, n := 0;
 init   etat=0, t=0;
 assert N=n;
        etat=1 => t <= 30;
        etat=2 => t <= 20; edon
```

Barrière

```
node GATE
 event  Go_down, Go_up, down, up;
 state  etat [0,3], y clock;
 trans  etat=0 |- Go_up -> ;
        etat=0 |- Go_down ->
              etat := 1, y := 0;
        etat=1 |- Go_down -> ;
        y <= 10 & etat=1 |- down ->
              etat := 2;
        etat=1 |- Go_up -> etat :=3, y := 0;
        etat=2 |- Go_down -> ;
        etat=2 |- Go_up  -> etat := 3, y :=0;
        etat=3 |- Go_up  -> ;
```

Barrière (II)

```
          etat=3 |- Go_down ->
                 etat := 1, y := 0;
          y <= 10 & etat=3 |- up -> etat := 0;
 init     etat=0, y = 0;
 assert   etat =1 => y <= 10;
          etat =3 => y <= 10;
edon
```

Contrôleur

```
node MAIN
 flow  N integer;
 event approach, exit, Go_up, Go_down;
 priorities   Go_up (<,k) approach;
 state etat [0,2], z clock;
 trans etat=0 |- approach -> etat := 1, z := 0;
       etat=0 & N>1 |- exit ->  ;
       etat=0 & N=1 |- exit -> etat := 2, z := 0;
       etat = 1 |- approach -> ;
       etat = 1 |- exit -> ;
       z <= 10 & etat=1 |- Go_down -> etat := 0;
       z <= 10 & etat=2 |- Go_up -> etat := 0;
       etat=2 |- approach -> etat := 1, z := 0;
```

Contrôleur (II)

```
sub t1, t2 : TRAIN, g : GATE;
sync
  <approach,t1.approach ?,t2.approach ?,-> >= 2;
  <Go_down,-,-,g.Go_down>;
  <exit,t1.exit ?,t2.exit ?,-> > 1;
  <Go_up,-,-,g.Go_up>;
init  etat = 0, z = 0;
assert  N=t1.N+t2.N;
        etat =1 => z <10;
        etat =2 => z <= 10;
edon
```

Plan

1. Systèmes temporisés
2. Exemple
3. AltaRica temporisé
4. Vérification temporelle

Ajout Syntaxe

Type horloge : dans les variables d'état et de flux

Garde : conjonction avec des contraintes sur les horloges

Affectation : reset sur les horloges

Assertion : invariant temporel $be \Rightarrow I$

Priorités : extension aux priorités temporelles. Si $e_1 <_k e_2$ alors e_1 ne sera tirable d'un état que si e_2 ne l'est pas dans un délai de k unités de temps.

Résultats

Priorités temporelles les 3 transitions standards sont modélisables en AltaRica temporisé

Sémantique d'AltaRica temporisé est donnée sous forme de système de transitions interfacé temporisé.

Algorithme de translation d'une description d'AltaRica temporisé en automate temporisé.

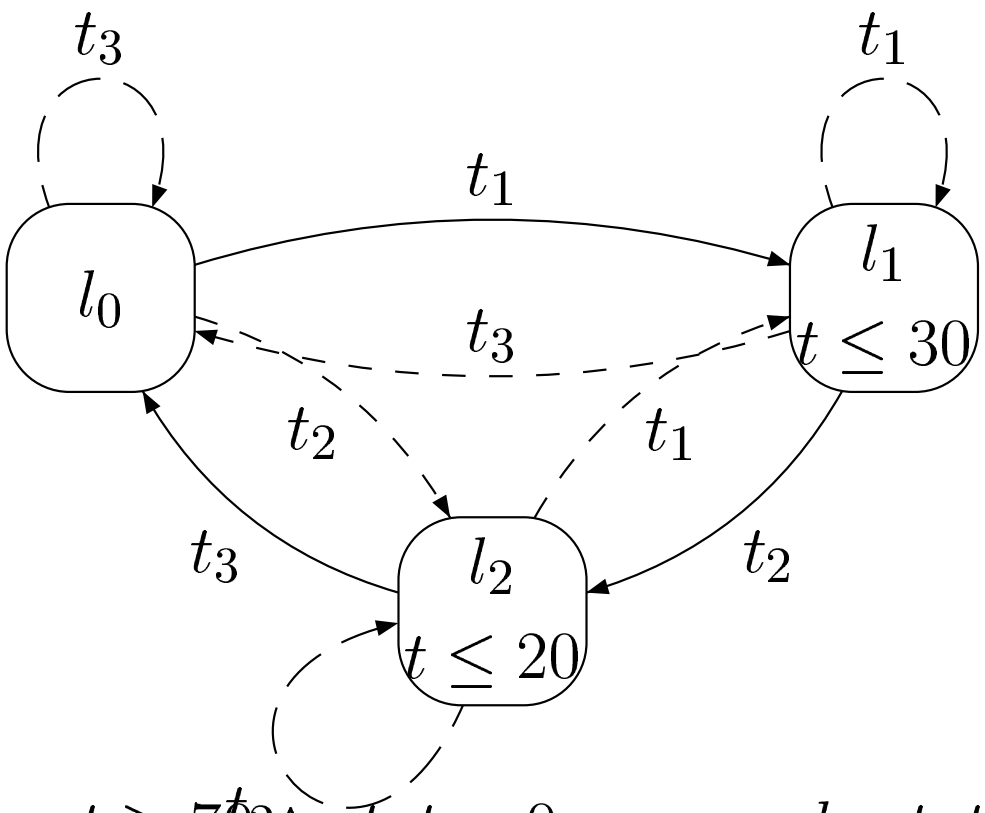
Translation en automates temporisés

Locations : décomposition de l'assertion en A_{V_T} et $A_{C_T} = \bigwedge_{j=1}^{p} P_j \implies I_j$. Alors on considère les locations :
$l_T^F = \left(\wedge_{j \in T} P_j\right) \bigwedge \left(\wedge_{j \in F} \neg P_j\right) \wedge A_{V_T}$

Invariants : conjonction des invariants $I(l_T^F) = \bigwedge_{k \in T} I_k$

Transitions : calcul de Pre pour chaque transition t $\mathit{Pre}_t(Q) = \{(s,f), \exists f', (a(s,f), f') \in Q\}$. Alors la relation de transitions :
$\forall l_T^F \in L, \forall t \in M, (l_T^F, (g \wedge \mathit{Pre}_t(l_{T'}^{F'}) \wedge \gamma, e, (a, R)), l_{T'}^{F'}) \in T$

Exemple train



t_1 $\quad t \geq 70 \wedge etat = 0, approach, etat := 1, t := 0, n := 1$

t_2 $\quad 20 \leq t \leq 30 \wedge etat = 1, in, etat := 2, t := 0$

t_3 $\quad 10 \leq t \leq 20 \wedge etat = 2, exit, etat := 0, t := 0, n := 0$

Plan

1. Systèmes temporisés
2. Exemple
3. AltaRica temporisé
4. Vérification temporelle

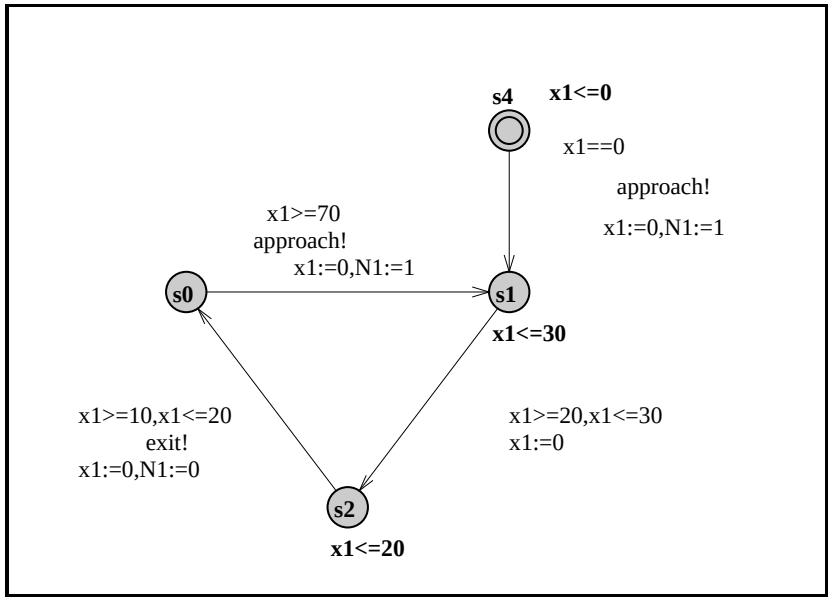
De l'intérêt des méthodes formelles

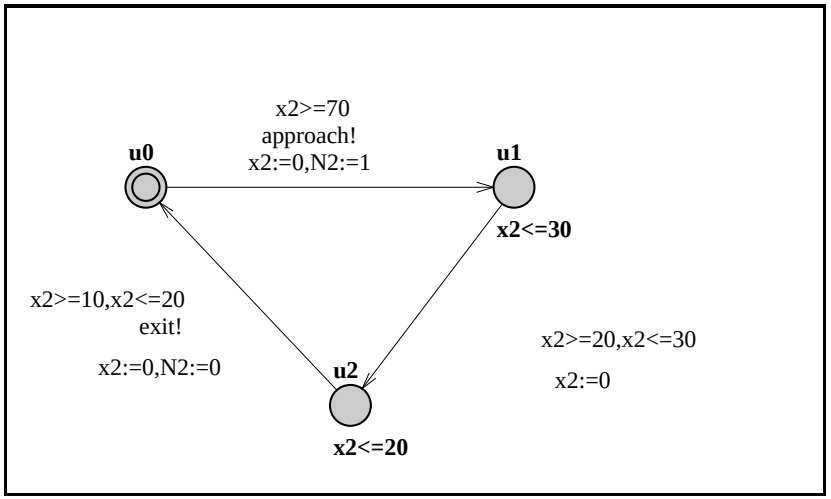
Méthodes formelles offrent un cadre mathématique strict qui lève certaines ambiguïtés et permet de vérifier certaines propriétés.

1. génération de test
2. démonstration automatique
3. model-checking

traduction $\rightarrow$ model checker UPPAAL

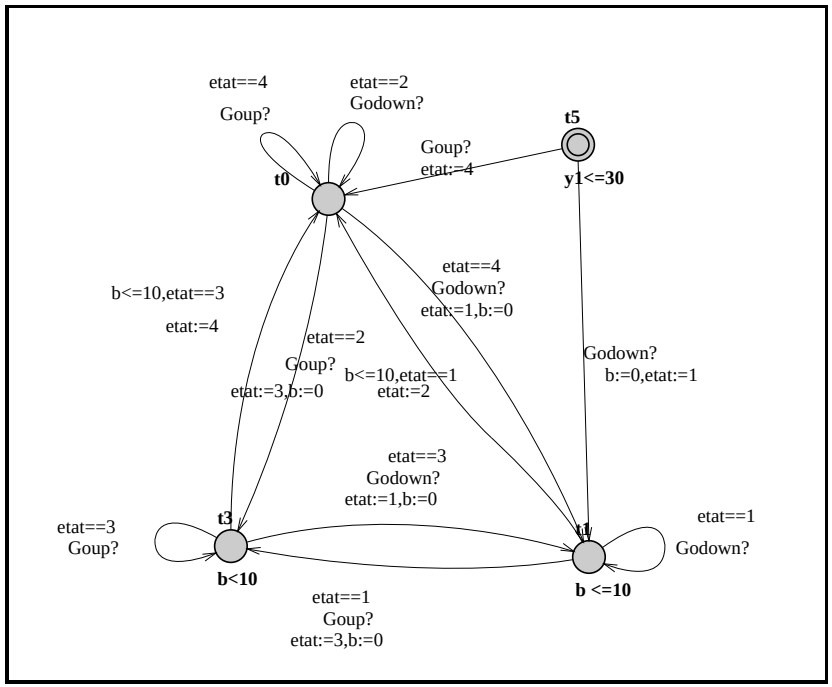
2 composants temporisés Train

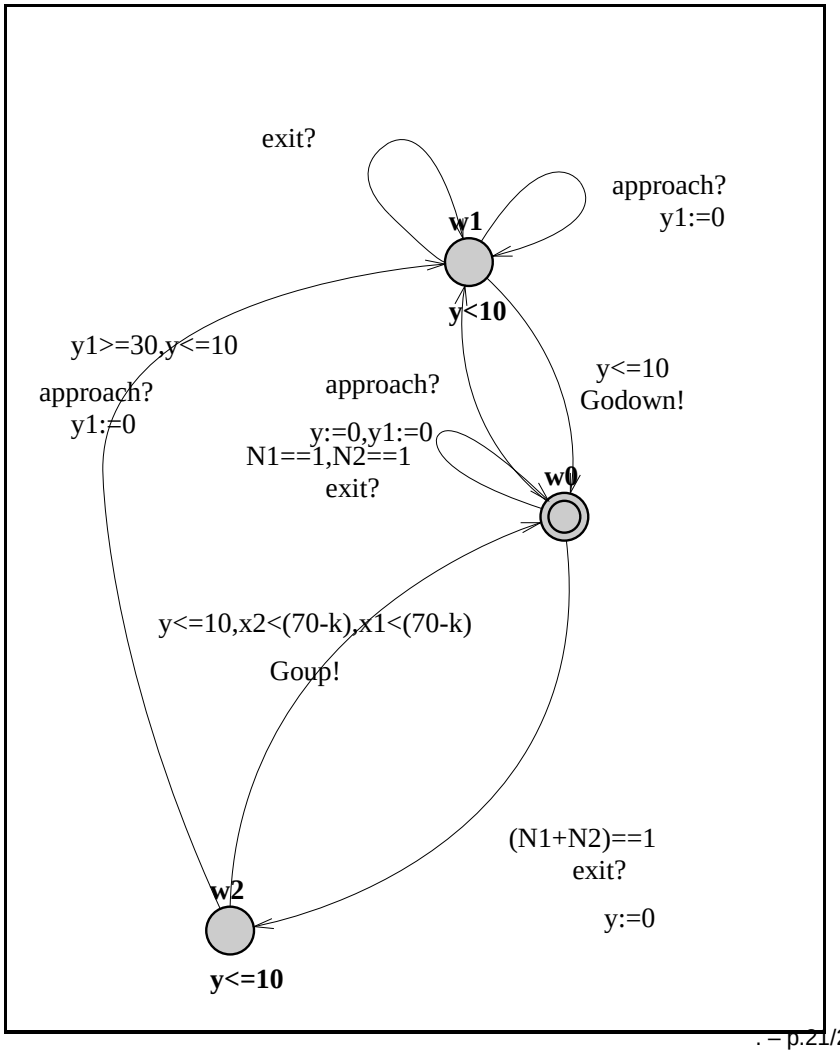




Etude de cas

Un contrôleur et une barrière





Résultats

On a introduit un paramètre k et on utilise UPPAAL pour analyser l'influence des priorités temporelles.

Sans temps : un train peut être en zone critique et la barrière ouverte

Temps sans priorité : propriété de sûreté vérifiée

Temps et priorité : il existe un chemin pour lequel la barrière ne se lève jamais ssi $k \geq 10$

Perspectives

Implémentation : ajouter une librairie de DBM (structure de données standard pour les horloges).
Coder le calcul des priorités temporelles.
Transformer en format UPPAAL

Autres extensions : automates hybrides et HYTECH

Vérification : hiérarchie (éviter la mise à plat)