Sorbonne Université
Paradigmes de Programmation Concurrente 51553

Cours 7 - Algèbres de processus temporels

Carlos Agon

15 octobre 2024

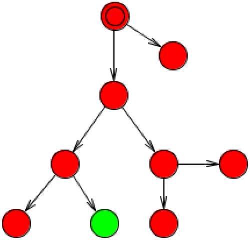
Un langage pour la vérification en Uppaal

On utilise une version simplifiée de TCTL (pour Temporal Computation Tree Logic)

- **Formules d'état** : pour une place p nous évaluons une condition pour verifier si elle est satisfait à chaque fois qu'on est dans cette place. Par exemple $S.p$ *and* $x > 3$ nous dit qu'à chaque fois qu'on est dans la place p du système S alors la valeur de x est plus grande que 3. Il y a une formule spéciale appelée *deadlock* qu'on peut appliquer à toute place.
- **Formules de chemin** :
  - Propriétés d'accessibilité
  - Propriétés de sécurité
  - Propriétés de vivacité
  - A,E,[ ],<>

Propriétés d'accessibilité

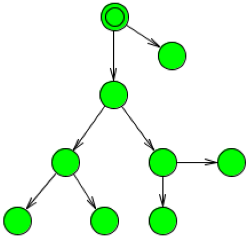
La question que l'on se pose est : est ce qu'une formule P peut être satisfaite ?
Ou y a-t-il un chemin partant de l_0 tq. P sera satisfaite à la fin du chemin ?



En UPPAAL on écrit `E<>P`

Propriétés de sécurité

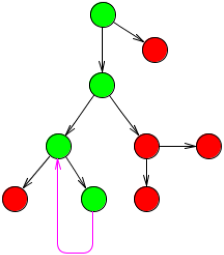
Les propriétés de sécurité sont de la forme : ''Quelque chose de mauvais ne doit jamais arriver''
En UPPAAL ces propriétés sont formulées positivement, c-à-d quelque chose bien est invariante.



On 'ecrit `A[]P`
A noter que `A[]P` = ¬ `E<>`¬ `P`

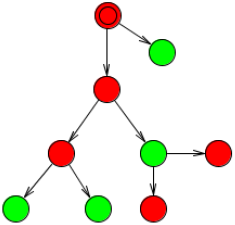
Propriétés de sécurité

`E[] P` dit qu'il doit exister un chemin maximale tq P est toujours vrai. Un chemin maximale est soit infini ou soit son dernier état n'a pas de transitions.

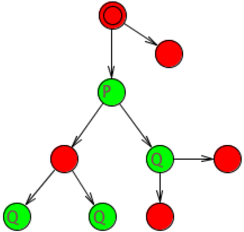


Propriétés de vivacité

Les propriétés de vivacité sont de la forme : "Quelque chose doit arriver"
P sera satisfait `A<> P` (`A<> P` = ¬ `E[]` ¬ `P`)



Une autre propriétés de vivacité utile est `P imply Q`



Un approche pragmatique

Définition

Un système temporel est un système avec une variable globale appellée **temps** qui a une influence sur les actions du système.

Hypotheses sur un système temporel :

- Il est composé de processus, chacun avec un **temps** propre. Chaque processus peut incrémenter sa variable **temps** pour faire progresser le temps.
- Il progresse de manière synchrone sur tous le processus : On increment le temps global de δ si tous le processus acceptent de le faire.
- Chaque pas d'exécution se fait en deux phases :
  - les processus exécutent des actions soit indépendamment soit en cooperation avec d'autres processus,
  - les processus se synchronisent pour faire avancer le temps.

Un nouvel pas dans l'execution commence uniquement quand la deuxième phase est finie (pour Lustre, Esterel, etc., chaque pas corresponde à une unitée de temps).

Plusieurs extensions pour une algèbre de processus

- ACP : Real Time ACP (Baeten et Bergstra, 91)
- ATP : Algebra of Time Processes (Sifakis et Nicollin, 90)
- TCSP : Timed CSP) (reed et Roscoe, 91)
- TeCCS : Temporal CCS (Moller et Tofts , 90)
- **TiCCS : Timed CCS (Wang Yi, 91)**
- TPL : Temporal Process Language (Hennessy et Regan, 91)
- U-LOTOS : Urgen LOTOS (Bolognesi, 91)

Le modèle

Le systèmes son modélisés par des LTS où :

- Les états sont des expressions de processus.
- Les étiquettes sont soit des actions $a \in A$ soit des $d \in D$ où D est un domain temporel. Les actions internes sont dénotées par τ

  $P \xrightarrow{a} Q$: le processus P réalise l'acction atemporelle a et après devient Q

  $P \xrightarrow{d} Q$: le processus P ne fait rien pendant un temps d et après devient Q

Domaine temporel

Un domaine temporel est donné par $(D, +, 0)$ satisfaisant :

- $d + d' = d \Leftrightarrow d' = 0$
- $d \leqslant d' \Leftrightarrow \exists d'' : d + d'' = d'$ ($\leqslant$ est un ordre total).
  0 est le plus petit element de D
  d'' est unique

Proprietés :

- D est dense si $\forall d, d' : d < d' \Rightarrow \exists d'' : d < d'' < d'$
- D est discret si $\forall d \exists d' : d < d' \wedge \forall d'' : d < d'' \Rightarrow d' \leqslant d''$
  d' est unique $d' = succ(d)$
- Exemples : $\mathbb{N}$ (discret), $\mathbb{R}^+$ (dense)

Le temps consideré est abstrait dans le sens qu'il est utilisé pour exprimer des contraintes entre les occurrences des actions.

Propriértés d'un modele

- Déterminisme temporel
- Additivitée temporelle
- Pas d'interblocage
- Action urgency
- Persistence
- Variabilité
- Contrôle borné

Quelles propriétés sont prioritaires et lesquelles on peut sacrifier ?

Déterminisme temporel

- $\forall P, P_1, P_2 \in \mathcal{P}$ et $\forall d \in \mathcal{D}_* : P \xrightarrow{d} P_1 \wedge P \xrightarrow{d} P_2 \Rightarrow P_1 = P_2$
  avec $\mathcal{D}_* = \mathcal{D} - \{0\}$

Déterminisme temporel

- $\forall P, P_1, P_2 \in \mathcal{P}$ et $\forall d \in \mathcal{D}_* : P \xrightarrow{d} P_1 \wedge P \xrightarrow{d} P_2 \Rightarrow P_1 = P_2$
  avec $\mathcal{D}_* = \mathcal{D} - \{0\}$

Le passage du temps est déterministe.
Si un processus P ne fait aucune action pendant un temps d alors son comportement est determiné uniquement par P et d.

Additivité du temps

- $\forall P, P' \in \mathcal{P}$ et $\forall d, d' \in \mathcal{D}_* : \exists P'' : P \xrightarrow{d} P'' \wedge P'' \xrightarrow{d'} P' \Leftrightarrow P \xrightarrow{d+d'} P'$

Additivité du temps

- $\forall P, P' \in \mathcal{P}$ et $\forall d, d' \in \mathcal{D}_* : \exists P'' : P \xrightarrow{d} P'' \wedge P'' \xrightarrow{d'} P' \Leftrightarrow P \xrightarrow{d+d'} P'$

Aussi appellée continuité.

- Si un processus peut attendre $d + d'$ alors il peut attendre d puis d' et vice-versa.

Dans le deux cas le comportement c'est le même.

Imposibilité d'interblockage

- $\forall P \in \mathcal{P}, \exists e \in A \cup \mathcal{D}_*, \exists P' \in \mathcal{P} : P \xrightarrow{e} P'$

Imposibilité d'interblockage

- $\forall P \in \mathcal{P}, \exists e \in A \cup \mathcal{D}_*, \exists P' \in \mathcal{P} : P \xrightarrow{e} P'$

Si on ne fait pas de difference entre terminaison et interblocage, alors, il n'y a pas d'état *sink* dans notre model.

Action urgency

Certains processus peuvent exécuter une action sans laisser passer le temps :

- $\exists P, P', a : P \xrightarrow{a} P' \wedge \forall d : P \overset{d}{\not\rightarrow}$

Dans TiCCS l'urgence est possible uniquement pour les actions invisibles :

- $\forall P, P', d : P \xrightarrow{\tau} P' \Rightarrow P \overset{d}{\not\rightarrow}$

Persistence

Le passage du temps ne peut pas supprimer la capacité de réaliser une action. Cette proprietée est appelée persistence.

- $\forall P, Q, d, a : P \xrightarrow{a} P' \wedge P \xrightarrow{d} Q \Rightarrow \exists P'' : Q \xrightarrow{a} P''$

Cette propriété n'est pas satisfaite par toutes les algèbres. Il existe de variantes comme la persistence d'intervalle.

- $\forall P\ \exists d > 0 \forall d' \in]0, d[, \forall Q, P', a : P \xrightarrow{d'} Q \wedge P \xrightarrow{a} P' \Rightarrow \exists P'' : Q \xrightarrow{a} P''$

Un processus garde la capacité de réaliser ses actions pendant une intervalle de temps uniquement.

Variabilité finie

- Un processus P a la propriété de *variabilité finie* s'il peut executer uniquement un ensemble fini d'actions dans une intervalle de temps finie.
  Pour assurer cette propriété on doit introduire un délai entre deux actions d'un processus sequential. Cela détruit l'abstraction temporelle.
- On definit la relation $\xRightarrow{(a,d)}$ par :
  $P \xRightarrow{(a,d)} R \iff P \xrightarrow{d} Q \wedge Q \xrightarrow{a} R$
  Et une *trace temporelle* de P par $(a_0, d_0)...(a_i, d_i)...$ tq. :
  $\exists P1, ..., P_i, ... : P \xRightarrow{(a_0,d_0)} P_1 ... \xRightarrow{(a_i,d_i)} P_{i+1} ...$
- $T(P)$ représente l'ensemble de traces de P

Variabilité bornée

- Un processus P a la propriété de *variabilité finie* ssi :
  $\forall d \, \forall \sigma = (a_0, d_0)...(a_i, d_i) \in T(P)$:
  $$\forall i, j (i < j \leqslant length(\sigma) \wedge \sum_{k=i+1}^{j} d_k \leqslant d) \Rightarrow j - i < \infty$$
- Un processus P a la propriété de *variabilité bornée* ssi :
  $\forall d \, \exists n \, \forall \sigma = (a_0, d_0)...(a_i, d_i) \in T(P)$:
  $$\forall i, j (i < j \leqslant length(\sigma) \wedge \sum_{k=i+1}^{j} d_k \leqslant d) \Rightarrow j - i \leqslant n$$

Contrôle bornée

- $init(P) = \{a | \exists P' : P \xrightarrow{a} P'\}$
- Un modele a la propriété de *controle borné* ssi :
  $\forall P, P', \exists d : si\ P \xrightarrow{d_1} P' \wedge init(P) \neq init(P')\ alors\ d \leqslant d_1$

Contrôle bornée

- $init(P) = \{a | \exists P' : P \xrightarrow{a} P'\}$
- Un modele a la propriété de *controle borné* ssi :
  $\forall P, P', \exists d : si\ P \xrightarrow{d_1} P' \wedge init(P) \neq init(P')\ alors\ d \leqslant d_1$

Si une place peut faire + ou - d'actions avec le cours du temps il est possible de trouver un temps borné *d* qui marque cette difference.

Temporal CCS

Syntaxe

Soit $\mathcal{N}$ ={a,b,c..} on définit Act $= \{a, \overline{a}, b, \overline{b}, c ...\} \cup \{\tau\}$
On définit l'ensemble de processus du CCS par :

- $D(\overrightarrow{x}) ::= P$
- $P ::= 0 \quad | \quad \alpha.P \quad | \quad D\langle\overrightarrow{v}\rangle \quad | \quad P + P \quad | \quad P|P \quad | \quad \nu a P$
- $\alpha ::= a|\overline{b}|\tau$

Extension

- Pour l'algèbre étendue nous avons Act $= \{a, \overline{a}, b, \overline{b}, c ...\} \cup \{\tau\} \cup D_*$
- L'ensemble d'opérateurs $\{|, +\}$ est étendus aussi avec des opérateurs temporels.

Deux principes à respecter

Un introduisant un domaine temporel D et un ensemble d'opérateurs temporels on aimerait garder les deux proprietés suivantes :

- **Conservation de la sémantique** : si on ne tient pas en compte les actions temporelles l'algèbre temporelle étendue doit se comporter comme CCS
- **Isomorphisme** : $\exists \overset{T}{\sim}$ tq. $P \sim Q$ dans CCS ssi $P \overset{T}{\sim} Q$ dans l'extension temporelle.

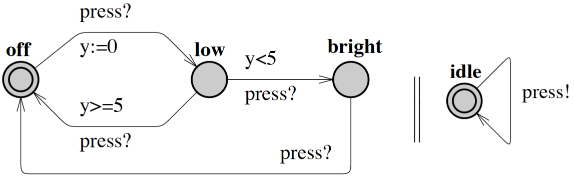
  En UPPAAL on a : $\dfrac{\forall u \in U,\ P \overset{u}{\nrightarrow}}{P \overset{d}{\rightarrow} P}$

  avec $U = \{$ urgent actions $\}$

Le modèle temporel d'UPPAAL

- D est dense, les horloges évaluent dans un nombre réel (un float)
- Un système est la mise en parallèle d'un ensemble d'automates temporels (TLST)
- Il y a possibilité d'horloges locales, mais tous les horloges progressent de manière synchrone.
- Un automate temporel est donné par (L, l_0, C, Act, E, I)
  - L est un ensemble des places et $l_0 \in L$, C est un ensemble d'horloges
  - $E \subseteq L \times A \times B(C) \times 2^C \times L$ defini les transitions avec
  - $B(C)$ un ensemble de gards
  - 2^C est un ensemble d'horloge à reseter
  - $I : L \to B(C)$ assigne des invariants aux places.

Exemple



(a) Lamp. (b) User.

$$(L, l_0, C, Act, E, I)$$

Sémantique

Soit un automate temporel (L, l_0, C, Act, E, I) la sémantique est donnée par un LTS $< S, s_0, \rightarrow >$ où

- $S \subseteq L \times \mathbb{R}^C$.
- $s_0 = (l_0, u_0)$ est l'état initial et
- $\rightarrow \subseteq S \times (\mathbb{R}_{\geqslant 0} \cup A) \times S$ est une relation de transition tq :
  - $(l, u) \xrightarrow{d} (l, u + d)$ si $\forall d' : 0 \leqslant d' \leqslant d \Rightarrow u + d' \in I(l)$
  - $(l, u) \xrightarrow{a} (l,' u')$ si $\exists\, e = (l, a, g, r, l') \in E$ tq. $u \in g, u' = 0$ si $u \in r$ et $u' \in I(l')$

Composants d'un système

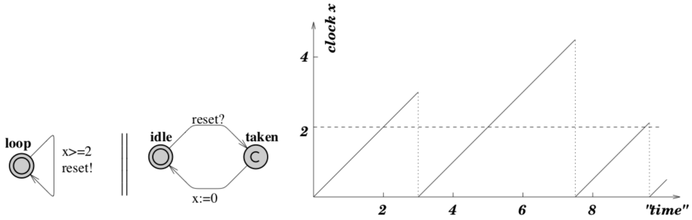
- **Processus** : des automates temporels.
- **Constantes et variables**
- **Synchronisation binaire** : par handshake
- **Broadcast channels** : *broadcast chan c*, quand on est dans une place p avec un canal c ! tous les templates qui sont dans un état q avec c? doivent se synchroniser. S'il n'y a pas détats avec c? alors p peut executer c!.
- **Urgent synchronisation** : si l'on declare le canal *urgent c* il ne peut pas y avoir des délai si une synchronisation est possible.
- **Urgent locations** : Le temps ne passe pas quand on arrive à ce type de places. C'est équivalant à avoir une horloge x qui est mise à zero sur toutes les transitions rentrantes et avoir une invariante $x <= 0$.
- **Commited locations** : Plus restrictif. S'il y a un ensemble de places marquées *committed* alors on doit obligatoirement choisir une au hazard et elle se comporte comme une place urgent.

Expressions sur le transitions et les places

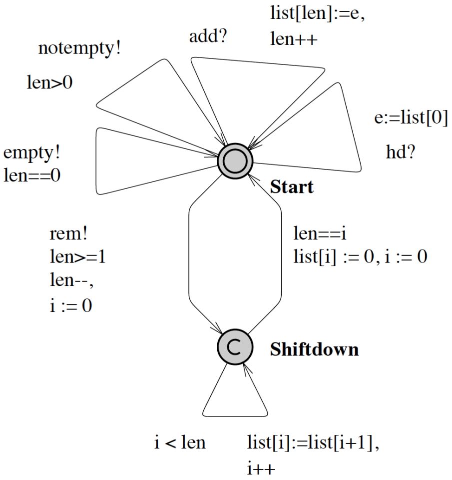
Sur les horloges et sur les integers

- **Select** : Uniquement sur une transition,
  name_1 : type_1, ... , name_n : type_n.
  Chaque variable prend une valoir aléatoire dans le range définit par le type.
- **Guard** : Uniquement sur une transition,
  Un prédicat sans effet de bord sur les horloges, variables et constantes.
  Elle doit être satisfaite si l'on veut s'engager dans la transition.
- **Update** : Uniquement sur une transition,
  x_1 : expr_1, ... , x_n, expr_n.
  x_i est assignée avec la valeur de expr_i
- **Invariant** : Uniquement sur une place,
  C'est une conjonction des conditions de la forme $x < e$ ou $x <= e$ où x est un horloge.

Démo



Exemple : gestion d'une file d'attente



Exemple : la même chose mais avec du code

```
const int N = 6;
typedef int[0,N-1] ele;

ele list[N+1];
int [0,N] len;

//mettre un element dans la queue
void enqueue (ele element) {
        list[len++] = element;
}

//return le premier element de la queue
void front () {
        return list[0];
}

//return le dernier element de la queue
void front () {
        return list[len - 1];
}

//remove le premier element de la queue
void dequeue () {
        int i = 0;
        len -= 1;
        while (i < len) {
                list[i] = list[i+1];
                i++
        }
        list[i] = 0;
}
```

Références

- Xavier Nicollin et Joseph Sifakis
  "An Overview and synthesis on Timed Process Algebras"
- Johan Bengtsson et Wang Yi
  "Timed Automata : Semantics, Algorithms and Tools"