

Sorbonne Université
Paradigmes de Programmation Concurrente 51553



Cours 5 - Algèbres de processus temporels

Carlos Agon

3 novembre 2020

Un approche pragmatique

Définition

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

Hypothèses 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 coopération avec d'autres processus,
 - les processus se synchronisent pour faire avancer le temps.

Un nouvel pas dans l'exécution commence uniquement quand la deuxième phase est finie (pour Lustre, Esterel, etc., chaque pas corresponde à une unité 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

Les systèmes sont 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 domaine temporel. Les actions internes sont dénotées par τ

$P \xrightarrow{a} Q$: le processus P réalise l'action 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 :

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

Propriétés :

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

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

Propriétés d'un modèle

- Déterminisme temporel
- Additivité 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 déterminé 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 appelée 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.

Impossibilité d'interblockage

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

Impossibilité d'interblocage

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

Si on ne fait pas de différence 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 :

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

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

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

Persistence

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

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

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

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

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 exécuter uniquement un ensemble fini d'actions dans un intervalle de temps finie.

Pour assurer cette propriété on doit introduire un délai entre deux actions d'un processus séquentiel. Cela détruit l'abstraction temporelle.

- On définit 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 $(a_0, d_0) \dots (a_i, d_i) \dots$ tq. :

$$\exists P_1, \dots, P_i, \dots : P \xrightarrow{(a_0, d_0)} P_1 \dots \xrightarrow{(a_i, d_i)} P_{i+1} \dots$$

- $T(P)$ représente l'ensemble de traces de P

Variabilité bornée

- Un processus P a la propriété de *variabilité finie* ssi :

$$\forall d \forall \sigma = (a_0, d_0) \dots (a_i, d_i) \in T(P) :$$

$$\forall i, j (i < j \leq \text{length}(\sigma) \wedge \sum_{k=i+1}^j d_k \leq d) \Rightarrow j - i < \infty$$

- Un processus P a la propriété de *variabilité bornée* ssi :

$$\forall d \exists n \forall \sigma = (a_0, d_0) \dots (a_i, d_i) \in T(P) :$$

$$\forall i, j (i < j \leq \text{length}(\sigma) \wedge \sum_{k=i+1}^j d_k \leq d) \Rightarrow j - i \leq n$$

Contrôle bornée

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

Contrôle bornée

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

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 différence.

Temporal CCS

Syntaxe

Soit $\mathcal{N} = \{a, b, c, \dots\}$ on définit $\text{Act} = \{a, \bar{a}, b, \bar{b}, c, \dots\} \cup \{\tau\}$

On définit l'ensemble de processus du CCS par :

- $D(\vec{x}) ::= P$
- $P ::= 0 \mid \alpha.P \mid D(\vec{v}) \mid P + P \mid P|P \mid \nu aP$
- $\alpha ::= a|\bar{b}|\tau$

Extension

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

Opérateurs temporels

- Delay
 - finite idling
 - time-stamped actions
 - Unbounded idling
- Urgence
 - actions immediates
 - time-stamped actions
 - asap
- Time-out
- Watchdog

Delay : finite idling

L'effet d'un délai est de retarder l'exécution d'un processus d'un paramètre d

L'opérateur finite idling $(d)P$ dit que P reste identique après d unités de temps.

$$(d)P \xrightarrow{d} P$$

$$\frac{d' < d}{(d)P \xrightarrow{d'} (d - d')P}$$

Delay : time-stamped actions

- La cas relatif $a[d]P$ exécute l'action a d unités de temps après la dernière action exécutée.

$$\frac{d' \leq d}{a[d]P \xrightarrow{d'} a[d - d']P}$$

$$a[0]P \xrightarrow{a} P$$

Delay : unbounded idling

Pour supprimer l'urgence des actions.

$$\frac{P \xrightarrow{a} P'}{\delta P \xrightarrow{a} P'} \quad \delta P \xrightarrow{d} \delta P$$

Si P ne peut pas exécuter l'action a
 δP sera au repos pour toujours.

Urgency : immediate actions

- immediate actions : $\dot{a}P$ impose que l'action a doit s'exécuter immédiatement

$$\dot{a}P \xrightarrow{a} P$$

- as soon as possible : dans quelques algèbres (e.g. U-LOTOS) on peut imposer l'urgence d'un certain nombre d'actions pendant l'exécution total d'un processus.

$$\frac{P \xrightarrow{a'} P'}{\text{asap}_a \text{ in } P \xrightarrow{a'} \text{asap}_a \text{ in } P'} \quad \frac{P \not\xrightarrow{a}, P \xrightarrow{d} P', \forall d' < d \forall Q, P \xrightarrow{d'} Q \Rightarrow Q \not\xrightarrow{a}}{\text{asap}_a \text{ in } P \xrightarrow{d} \text{asap}_a \text{ in } P'}$$

Le deuxième règle dit que $\text{asap}_a \text{ in } P$ peut rester idle d unités de temps si P peut le faire et ne s'engage pas avec a avant d unités de temps.

Timeout

- Un opérateur timeout est composé de $P(\text{body}), Q(\text{exception}) \in \mathcal{P}$ et de $d \in \mathcal{D}_*$.
- $P \triangleright^d Q$ se comporte comme P si une action initiale de P est exécutée dans un temps d sinon il se comporte comme Q après un temps d .
- Beaucoup d'interprétations par rapport à **action initiale, dans un temps d et après un temps d**

$$\begin{array}{c}
 \frac{P \xrightarrow{a} P'}{P \triangleright^d Q \xrightarrow{a} P'} \\
 \\
 \frac{P \xrightarrow{d} P'}{P \triangleright^d Q \xrightarrow{d} Q} \\
 \\
 \frac{P \xrightarrow{d'} P', d' < d}{P \triangleright^d Q \xrightarrow{d'} P' \triangleright^{d-d'} Q} \\
 \\
 \frac{P \xrightarrow{d} P', Q \xrightarrow{d'} Q'}{P \triangleright^d Q \xrightarrow{d+d'} Q'}
 \end{array}$$

Deux principes à respecter

Un introduisant un domaine temporel D et un ensemble d'opérateurs temporels on aimerait garder les deux propriétés 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** : $\exists \sim^T$ tq. $P \sim Q$ dans CCS ssi $P \sim^T Q$ dans l'extension temporelle.

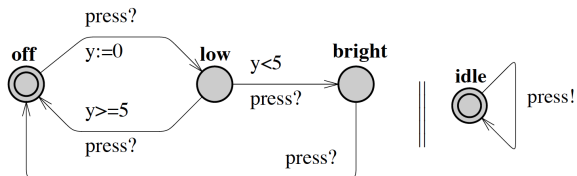
$$\text{En UPPAAL on a : } \frac{\forall u \in U, P \xrightarrow{u}}{P \xrightarrow{d} P}$$

avec $U = \{ \text{urgents actions} \}$

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
 - $E \subseteq L \times A \times B(C) \times 2^C \times L$ défini les transitions avec $B(C)$ un ensemble de guards
 - $I : L \rightarrow B(C)$ assigne des invariants aux places.
 - 2^C est un ensemble d'horloge à reseter.

Example



(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 $\langle S, s_0, \rightarrow \rangle$ où

- $S \subseteq L \times \mathbb{R}^C$.
- $s_0 = (l_0, u_0)$ est l'état initial et
- $\rightarrow \subseteq S \times (\mathbb{R}_{\geq 0} \cup A) \times S$ est une relation de transition tq :
 - $(l, u) \xrightarrow{d} (l, u + d)$ si $\forall d' : 0 \leq d' \leq d \Rightarrow u + d' \in I(l)$
 - $(l, u) \xrightarrow{a} (l', u')$ 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 d'états avec c alors p peut exécuter c !.
- **Urgent synchronisation** : si l'on déclare le canal *urgent c* il ne peut pas y avoir de délai si une synchronisation est possible.
- **Urgent locations** : Le temps ne passe pas quand on arrive à ce type de places. C'est équivalent à avoir une horloge x qui est mise à zéro sur toutes les transitions entrantes et avoir une invariante $x \leq 0$.
- **Committed locations** : Plus restrictif. S'il y a un ensemble de places marquées *committed* alors on doit obligatoirement choisir une au hasard et elle se comporte comme une place urgente.

Expressions sur les transitions et les places

Sur les horloges et sur les entiers

- **Select** : Uniquement sur une transition,
 $\text{name}_1 : \text{type}_1, \dots, \text{name}_n : \text{type}_n$.
Chaque variable prend une valeur aléatoire dans le range défini 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,
 $x_1 : \text{expr}_1, \dots, x_n, \text{expr}_n$.
 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 $x \leq e$ où x est un horloge.

Un langage pour la vérification

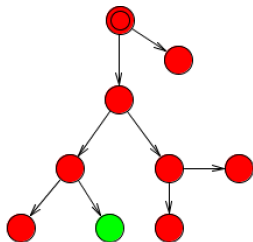
On utilise une version simplifiée de TCTL (pour Temporal Computation Tree Logic)

- **Formules d'état** : pour une place p nous évaluons une condition pour vérifier si elle est satisfaite à chaque fois qu'on est dans cette place. Par exemple $P.l \text{ and } x > 3$ nous dit qu'à chaque fois qu'on est dans la place l du processus P alors la valeur de l'horloge x est plus grande que 3.
Il y a une formule spéciale appelée *deadlock*
- **Formules de chemin** :
 - Propriétés d'accessibilité
 - Propriétés de sécurité
 - Propriétés de vivacité

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 ?

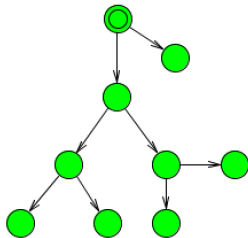


En UPPAAL on écrit $E \langle \rangle P$

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.

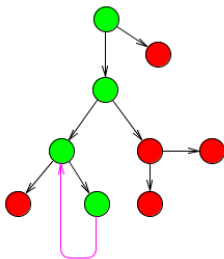


On 'ecrit $A[]P$

A noter que $A[]P = \neg E\langle\rangle\neg P$

Propriétés de sécurité

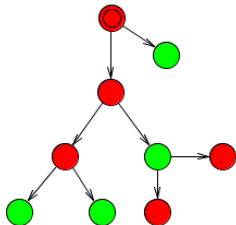
$E \Box P$ dit qu'il doit exister un chemin maximale tq P est toujours vrai.
Un chemin maximale est soit infini ou soit son dernier état n'a pas de transitions.



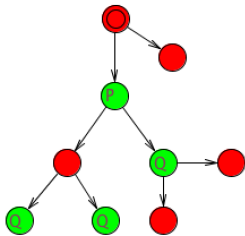
Propriétés de vivacité

Les propriétés de vivacité sont de la forme : "Quelque chose doit arriver"

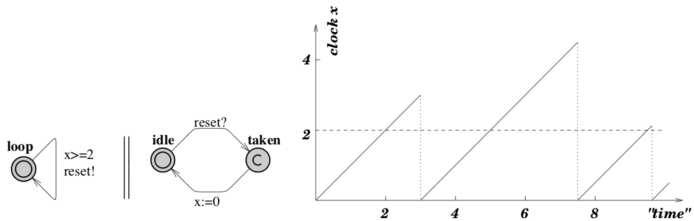
P sera satisfait $A \langle \rangle P$ ($A \langle \rangle P = \neg E[] \neg P$)



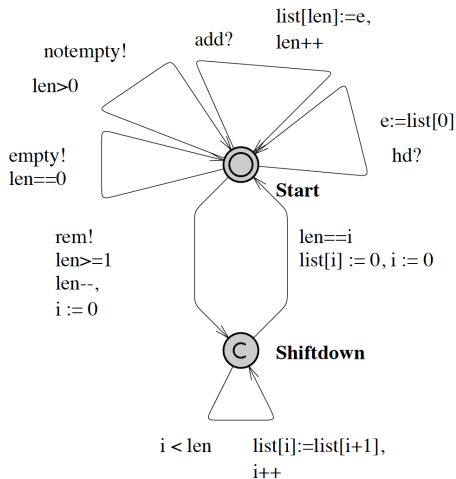
Une autre propriétés de vivacité utile est $P \text{ imply } Q$



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”