

Systèmes de transitions

Philippe Quéinnec, Xavier Thirioux

Département Informatique et Mathématiques appliquées
ENSEEIH

26 janvier 2018



Méthodes formelles ?

Pourquoi ?

- Nécessité de **prouver** qu'un programme possède bien les propriétés attendues
- C'est dur \Rightarrow nécessité de cadres formels précis et d'outils

Comment ?

- Langage classique : état = valeurs des variables + *flot de contrôle implicite*
- Système de transitions : état = valeurs des variables.
flot de contrôle *explicite*

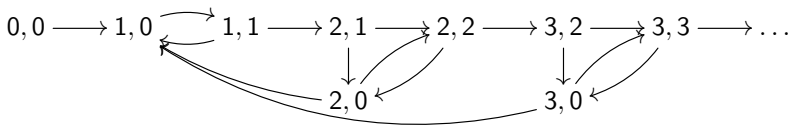


Exemple

Soit trois processus exécutant concurremment (par entrelacement) :

boucle $x \leftarrow y + 1$ || boucle $y \leftarrow x$ || boucle $y \leftarrow 0$

① Description du système en termes d'états ?



② Propriétés :

- L'état 4,2 est-il accessible ?
- Le système s'arrête-t-il ? Toujours, parfois ?
- Est-il toujours vrai que $y = 0 \vee 0 \leq x - y \leq 1$?
- Si $y = 6$, est-il possible/nécessaire que x devienne > 6 ?
- Est-il possible/nécessaire que y soit non borné ?

Temporal Logic of Actions

- ① Un langage de spécification logique (LTL / Logique temporelle linéaire) \approx quelles sont les propriétés attendues
- ② Un langage d'actions \approx un langage de spécification plus opérationnel \approx un langage de programmation
- ③ (en fait, langage de spécification = langage d'actions)
- ④ Cadre formel = système de transitions

Auteurs : **Leslie Lamport**, Martín Abadi



Plan du cours

- 1 Systèmes de transitions
- 2 TLA⁺ : les actions
- 3 Équité dans les systèmes de transitions
- 4 Logique temporelle linéaire LTL
- 5 TLA⁺ : la logique et l'équité
- 6 Logique temporelle arborescente CTL



Première partie

Systèmes de transitions



Introduction

Objectifs

Représenter les exécutions d'un programme en faisant abstraction de certains détails :

- les détails sont la cause d'une explosion du nombre d'états et de la complexité des traitements.
- ne conserver que ce qui est pertinent par rapport aux propriétés attendues.



Utilisation

Un système de transitions peut être construit :

- *avant* l'écriture du programme, pour explorer la faisabilité de celui-ci.
Le programme final est un raffinement en utilisant le système de transitions comme guide.
- *après* l'écriture du programme, par abstraction, en ne conservant que les aspects significatifs du programme réel.



Plan

- 1 Définitions
 - Système de transitions
 - Traces, exécutions
 - États, graphe
 - Système de transitions étiqueté
- 2 Représentations
 - Explicite
 - Implicite
- 3 Propriétés générales
 - Déterminisme
 - Blocage
 - Réinitialisable
 - Bégaiement



Système de transitions

ST

Un système de transitions est un triplet $\langle S, I, R \rangle$.

- S : ensemble d'états. Peut être fini ou infini.
- $I \subseteq S$: ensemble des états initiaux.
- $R \subseteq S \times S$: relation (de transitions) entre paires d'états.
 $(s, s') \in R$ signifie qu'il existe une transition faisant passer le système de l'état s à l'état s' .

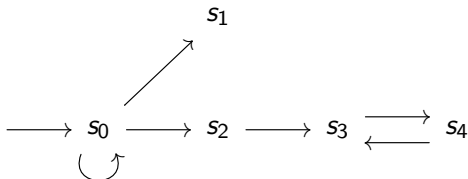


Exemple - système de transitions

$$S = \{s_0, s_1, s_2, s_3, s_4\}$$

$$I = \{s_0\}$$

$$R = \{(s_0, s_0), (s_0, s_1), (s_0, s_2), (s_2, s_3), (s_3, s_4), (s_4, s_3)\}$$



Séquences

Séquence

Soit S un ensemble.

$S^* \triangleq$ l'ensemble des séquences finies sur S .

$S^\omega \triangleq$ l'ensemble des séquences infinies sur S .

$\sigma_i \triangleq$ le $i^{\text{ème}}$ (à partir de 0) élément d'une séquence σ .

Conventions de représentation :

- Une séquence s est notée sous la forme : $\langle s_1 \rightarrow s_2 \rightarrow \dots \rangle$.
- $\langle \rangle$: la séquence vide.

Pour une séquence finie σ :

- $\sigma^* \triangleq$ l'ensemble des séquences finies produites par la répétition arbitraire de σ .
- $\sigma^+ \triangleq \sigma^* \setminus \{\langle \rangle\}$
- $\sigma^\omega \triangleq$ la séquence infinie produite par la répétition infinie de σ .

Traces

Traces finies

Soit $\langle S, I, R \rangle$ un système de transitions.

On appelle **trace finie** une séquence finie $\sigma \in S^*$ telle que :

- $\sigma = \langle s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_{n-1} \rightarrow s_n \rangle$
- $\forall i \in [0..n[: (s_i, s_{i+1}) \in R$

Traces finies maximales

Soit $\langle S, I, R \rangle$ un système de transitions.

Une trace finie $\langle s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_{n-1} \rightarrow s_n \rangle \in S^*$ est **maximale** ssi il n'existe pas d'état successeur à s_n , i.e. $\forall s \in S : (s_n, s) \notin R$.

Une trace maximale va le plus loin possible.



Traces infinies

Traces infinies

Soit $\langle S, I, R \rangle$ un système de transitions, et $s_0 \in S$.

On appelle **trace infinie** à partir de s_0 un élément $tr \in S^\omega$ tel que :

- $tr = \langle s_0 \rightarrow s_1 \rightarrow s_2 \dots \rangle$
- $\forall i \in \mathbb{N} : (s_i, s_{i+1}) \in R$

Traces

Soit $\langle S, I, R \rangle$ un système de transitions, et $s \in S$.

$Traces(s) \triangleq$ l'ensemble des traces infinies ou finies maximales commençant à l'état s .

Exécutions

Exécutions

Soit $\mathcal{S} = \langle S, I, R \rangle$ un système de transitions.

Une **exécution** $\sigma = \langle s_0 \rightarrow \dots \rangle$ est une trace infinie ou finie maximale telle que $s_0 \in I$.

$Exec(\mathcal{S}) \triangleq$ l'ensemble des exécutions de \mathcal{S} .

On a une (seule et unique) exécution vide $\langle \rangle$ ssi $I = \emptyset$.

Préfixe d'exécution

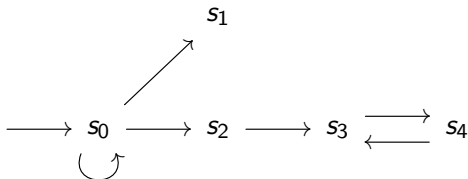
Pour σ une exécution, un préfixe d'exécution est une trace finie

$\sigma_p = \langle s_0 \rightarrow \dots \rightarrow s_n \rangle$ telle que $s_0 \in I$ et $\sigma = \sigma_p \rightarrow \dots$

s_n est appelé l'état final de ce préfixe.



Exemple - traces



$s_0 \rightarrow s_0 \rightarrow s_2 \rightarrow s_3$ est une trace finie non maximale

$$\text{Traces}(s_1) = \langle s_1 \rangle$$

$$\text{Traces}(s_3) = \langle (s_3 \rightarrow s_4)^\omega \rangle$$

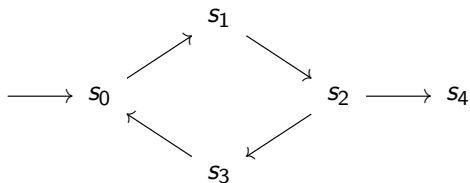
$$\text{Traces}(s_2) = \langle s_2 \rightarrow (s_3 \rightarrow s_4)^\omega \rangle$$

$$\text{Traces}(s_0) = \langle s_0^\omega \rangle, \langle s_0^+ \rightarrow s_1 \rangle, \langle s_0^+ \rightarrow s_2 \rightarrow (s_3 \rightarrow s_4)^\omega \rangle$$

$$\text{Exec}(\mathcal{S}) = \text{Traces}(s_0)$$



Exemple 2 - traces, exécutions

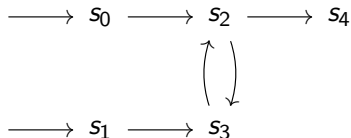


$Traces(s_2) =$

$Traces(s_0) =$

$Exec(\mathcal{S}) =$

Exemple 3 - traces, exécutions



$$\text{Traces}(s_2) =$$

$$\text{Traces}(s_0) =$$

$$\text{Traces}(s_1) =$$

$$\text{Exec}(\mathcal{S}) =$$



États accessibles

État accessible

Soit $\mathcal{S} = \langle S, I, R \rangle$ un système de transitions.

$s \in S$ est un état **accessible** ssi il existe un préfixe d'exécution dont l'état final est s .

$Acc(\mathcal{S}) \triangleq$ l'ensemble des états accessibles de \mathcal{S} .



États récurrents

Ensemble d'état récurrent

Soit $\mathcal{S} = \langle S, I, R \rangle$ un système de transitions et $\sigma = \langle s_0 \rightarrow \dots \rangle$ une exécution.

Un ensemble d'états P est **récurrent** dans σ si :

- cas σ infinie : $\forall i \in \mathbb{N} : \exists j \geq i : s_j \in P$
(P apparaît une infinité de fois dans σ).
- cas σ finie : l'état final de σ est dans P .

$\text{Inf}_{\mathcal{S}}(P, \sigma) \triangleq P$ est un ensemble d'états récurrent dans σ .

Note : on dit aussi *infiniment souvent présent* dans σ .



Transitions récurrentes

Ensemble de transitions récurrent

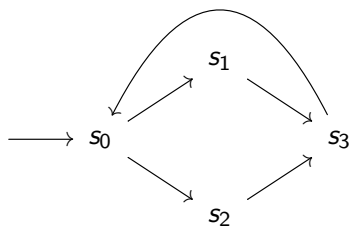
Soit $\mathcal{S} = \langle S, I, R \rangle$ un système de transitions et $\sigma = \langle s_0 \rightarrow \dots \rangle$ une exécution.

Un ensemble de transitions Q est **récurrent** dans σ si :

- cas σ infinie : $\forall i \in \mathbb{N} : \exists j \geq i : s_j \rightarrow s_{j+1} \in Q$
(des transitions de Q apparaissent une infinité de fois dans σ).
- cas σ finie : la transition finale de σ est dans Q
($\sigma = \langle s_0 \rightarrow \dots \rightarrow s \rightarrow s' \rangle \wedge s \rightarrow s' \in Q$).

$\text{Inf}_{\mathcal{T}}(Q, \sigma) \stackrel{\Delta}{=} Q$ est un ensemble de transitions récurrent dans σ .

Exemple - états récurrents



s_1 récurrent dans $\langle (s_0 \rightarrow s_1 \rightarrow s_3)^\omega \rangle$
 s_1 récurrent dans $\langle (s_0 \rightarrow s_1 \rightarrow s_3 \rightarrow s_0 \rightarrow s_2 \rightarrow s_3)^\omega \rangle$
 s_1 pas récurrent dans $\langle (s_0 \rightarrow s_1 \rightarrow s_3)^* \rightarrow (s_0 \rightarrow s_2 \rightarrow s_3)^\omega \rangle$

$s_1 \rightarrow s_3$ récurrente dans $\langle (s_0 \rightarrow s_1 \rightarrow s_3 \rightarrow s_0 \rightarrow s_2 \rightarrow s_3)^\omega \rangle$
 $s_1 \rightarrow s_3$ pas récurrente dans $\langle (s_0 \rightarrow s_1 \rightarrow s_3)^* \rightarrow (s_0 \rightarrow s_2 \rightarrow s_3)^\omega \rangle$

Graphe des exécutions

Graphe des exécutions

Soit $\mathcal{S} = \langle S, I, R \rangle$ un système de transitions.

Le **graphe des exécutions** est le graphe orienté où :

- l'ensemble des sommets est $Acc(\mathcal{S})$;
- l'ensemble des arêtes orientées est R , restreint aux seuls états accessibles.

Il s'agit donc du graphe $\langle S \cap Acc(\mathcal{S}), R \cap (Acc(\mathcal{S}) \times Acc(\mathcal{S})) \rangle$.



Système de transitions étiqueté

ST étiqueté

Un système de transitions étiqueté est un quintuplet $\langle S, I, R, L, Etiq \rangle$.

- S : ensemble d'états.
- $I \subseteq S$: ensemble des états initiaux.
- $R \subseteq S \times S$: relation de transitions entre paires d'états.
- L : ensemble d'étiquettes.
- $Etiq$: fonction qui associe une étiquette à chaque transition :
 $Etiq \in R \rightarrow L$.

Un ST étiqueté se rapproche beaucoup des automates.
Mais : pas d'état terminal + exécution infinie.



Équivalence aux ST sans étiquette

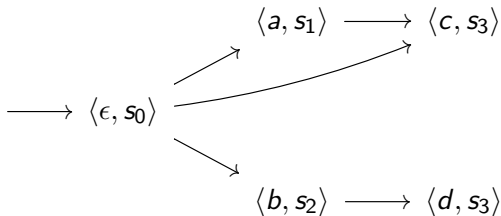
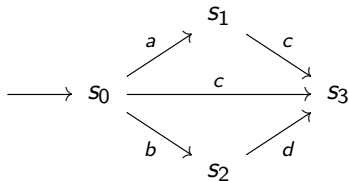
Un système de transitions étiqueté $\langle S, I, R, L, Etiq \rangle$ est équivalent au système sans étiquette $\langle S', I', R' \rangle$ défini par :

- $S' = (L \cup \{\epsilon\}) \times S$
- $I' = \{\epsilon\} \times I$
- $R' = \{(\langle I, s \rangle, \langle I', s' \rangle) \mid (s, s') \in R \wedge I' = Etiq(s, s')\}$

Une transition $s_1 \xrightarrow{a} s_2$ devient $\langle _, s_1 \rangle \longrightarrow \langle a, s_2 \rangle$, où $_$ est n'importe quelle étiquette.



Exemple - équivalence avec/sans étiquette



Différences avec un automate

Système de transitions \neq automate

- Pas d'étiquette sur les transitions (ou comme si)
- Une transition n'est pas causée par l'environnement
- Pas d'états terminaux
- Nombre d'états infini possible
- Exécution infinie possible



Plan

- 1 Définitions
 - Système de transitions
 - Traces, exécutions
 - États, graphe
 - Système de transitions étiqueté
- 2 Représentations
 - Explicite
 - Implicite
- 3 Propriétés générales
 - Déterminisme
 - Blocage
 - Réinitialisable
 - Bégaiement



Représentation en extension

Donnée en extension du graphe des exécutions, par exemple sous forme graphique ou par l'ensemble des sommets et arêtes.
Ne convient que pour les systèmes de transitions où le nombre d'états et de transitions est fini.



Représentation en intention

Représentation symbolique à l'aide de variables.

Système de transitions à base de variables

Triplet $\langle V, Init, Trans \rangle$ où

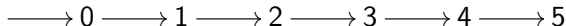
- $V = \{v_i\}_{i \in I}$: ensemble fini de variables.
- $Init(\{v_i\}_{i \in I})$: prédicat définissant les états initiaux et portant sur les variables v_i .
- $Trans(\{v_i, v'_i\}_{i \in I})$: prédicat définissant les transitions, portant sur les variables v_i représentant l'état courant et les variables v'_i représentant l'état suivant.



Exemple : un compteur borné

```
i = 0;
while (i < N) {
  i = i+1;
}
```

Graphe d'exécution pour $N = 5$.



Symboliquement :

$$V \triangleq i \in \mathbb{N}$$

$$I \triangleq i = 0$$

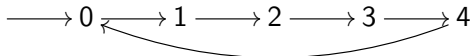
$$R \triangleq i < N \wedge i' = i + 1 \quad \text{ou} \quad R \triangleq i' \leq N \wedge i' - i = 1$$



Exemple : un compteur cyclique

```
i = 0;
while (true) {
  i = (i+1) % N;
}
```

Graphe d'exécution pour $N = 5$.



Symboliquement :

$$V \triangleq i \in \mathbb{N}$$

$$I \triangleq i = 0$$

$$R \triangleq i' = (i + 1) \bmod N$$



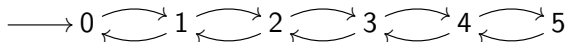
Exemple : un entier oscillant

```

i = 0;
while (true) {
    i > 0 -> i = i - 1;
    or
    i < N -> i = i + 1;
}

```

Graphe d'exécution pour $N = 5$.



Symboliquement :

$$V \triangleq i \in \mathbb{N}$$

$$I \triangleq i = 0$$

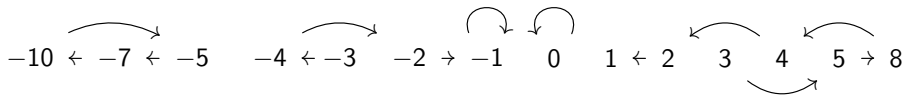
$$R \triangleq i > 0 \wedge i' = i - 1 \quad \text{ou} \quad R \triangleq |i' - i| = 1 \wedge 0 \leq i' \leq N$$

$$\vee i < N \wedge i' = i + 1$$

Exemple : le problème de Collatz

```
void Collatz(int n) {
  while (n != 1)
    if (n mod 2 = 0) n = n / 2; else n = (3*n + 1) / 2;
}
```

Graphe d'exécution pour $-5 \leq n \leq 5$ initialement.



Symboliquement :

$$V \triangleq n \in \mathbb{N}$$

$$I \triangleq -5 \leq n \leq 5$$

$$R \triangleq n \neq 1 \wedge \left(\begin{array}{l} (n' = n/2 \wedge n \equiv 0[2]) \\ \vee (n' = (3n+1)/2 \wedge n \equiv 1[2]) \end{array} \right)$$

ST correspondant

Le système de transitions correspondant est $\langle S, I, R \rangle$ où :

- $S = \prod_{i \in I} D_i$, où $\{D_i\}_{i \in I}$ sont les domaines (types) de $\{v_i\}_{i \in I}$
- $I = \text{Init}(\{v_i\}_{i \in I})$
- $R = \text{Trans}(\{v_i, v'_i\}_{i \in I})$



Prédicats

Prédicat d'état

Un prédicat d'état est un prédicat portant sur les variables (d'état) d'un système donné en intention.

Un prédicat d'état peut être vu comme la fonction caractéristique d'une partie de S .

Prédicat de transition

Un prédicat de transitions est un prédicat portant sur les variables (d'état) primées et non primées.

Un prédicat de transitions peut être vu comme la fonction caractéristique d'une partie de $S \times S$.



Exemple - prédicats

$$V \triangleq n \in \mathbb{N}$$

$$I \triangleq -5 \leq n \leq 5$$

$$R \triangleq n \neq 1 \wedge \left(\begin{array}{l} (n' = n/2 \wedge n \equiv 0[2]) \\ \vee (n' = (3n+1)/2 \wedge n \equiv 1[2]) \end{array} \right)$$

Prédicats d'état : $I, n < 20$

Prédicats de transition : $R, n' - n > 3$



Plan

- 1 Définitions
 - Système de transitions
 - Traces, exécutions
 - États, graphe
 - Système de transitions étiqueté
- 2 Représentations
 - Explicite
 - Implicite
- 3 Propriétés générales
 - Déterminisme
 - Blocage
 - Réinitialisable
 - Bégaïement



Déterminisme

Déterminisme fort

Un système est déterministe fort \triangleq la relation de transition est fonctionnelle, i.e. $\forall s \in S : |\{s' \in S : (s, s') \in R\}| \leq 1$.

Déterminisme opérationnel

Un système est déterministe au sens opérationnel \triangleq la relation de transition *restreinte aux états accessibles* est fonctionnelle.

Note : ST étiqueté déterministe \neq automate déterministe
(les étiquettes ne comptent pas)



Déterminisme – exemple

$$\begin{aligned}
 V &\triangleq n \in \mathbb{N} \\
 I &\triangleq n = 0 \\
 R &\triangleq \begin{aligned} &n \leq 4 \wedge n' = n + 1 \\ &\vee n \% 3 = 0 \wedge n' - n = 1 \\ &\vee n \geq 6 \wedge n' = n + 2 \end{aligned}
 \end{aligned}$$

Graphe d'exécution : $\rightarrow 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \quad 5 \quad 6 \rightarrow 7 \dots$
 \searrow
 $8 \dots$

Déterministe opérationnellement, mais pas fortement déterministe.



Blocage

Interblocage

Un système possède un interblocage (deadlock) \triangleq il existe un état accessible sans successeur par la relation R .

De manière équivalente un système possède un interblocage s'il existe des exécutions finies.

Pour les systèmes modélisant des programmes séquentiels classiques, l'interblocage est équivalent à la terminaison.

Boucle

Un système possède une boucle (livelock) \triangleq il existe un état accessible dont le seul successeur par la relation R est lui-même.



Réinitialisable

Réinitialisable

Un système est réinitialisable \triangleq depuis tout état accessible, il existe une trace finie menant à un état initial.

Cette propriété signifie qu'à n'importe quel moment, il existe une séquence d'actions pour revenir à l'état initial du système et ainsi redémarrer.



Bégaïement

Bégaïement

Un système de transitions bégaie ssi tout état possède une boucle vers lui-même : $R \supseteq Id$.

Utilité :

- Modéliser l'avancement arbitraire : $\longrightarrow \rightarrow s_0 \xrightarrow{\text{boucle}} s_0 \longrightarrow s_1$

on peut aller en s_1 après être resté arbitrairement longtemps en s_0 .
- N'avoir que des exécutions infinies : tout état sans successeur (dans un système sans bégaïement) a un unique successeur avec bégaïement : lui-même. La terminaison (l'interblocage) $\dots \rightarrow s_i$ est alors $\dots \rightarrow s_i^\omega$.
- Composer plusieurs systèmes de transitions.



Composition de systèmes de transitions

Composition

La composition des ST avec bégaiement $\langle V_1, I_1, R_1 \rangle$ et $\langle V_2, I_2, R_2 \rangle$ est $\langle V, I, R \rangle$ où :

- $V \triangleq V_1 \cup V_2$ (union des variables)
- $I \triangleq I_1 \wedge I_2$ (les deux sous-systèmes démarrent dans un état initial respectif)
- $R \triangleq R_1 \wedge R_2$ (les deux sous-systèmes évoluent chacun selon ses transitions)

Comme R_1 et R_2 peuvent bégayer, $R_1 \wedge R_2$ signifie donc qu'on peut exécuter une transition de R_1 seule et R_2 bégayant, ou bien réciproquement, ou bien encore exécuter R_1 en même temps que R_2 .



Exemple - composition

$$\left(\begin{array}{l} V_1 \triangleq i \in \mathbb{N} \\ l_1 \triangleq i = 0 \\ R_1 \triangleq i' = i + 1 \\ \quad \vee i' = i \end{array} \right) \otimes \left(\begin{array}{l} V_2 \triangleq j \in \mathbb{N} \\ l_2 \triangleq j = 0 \\ R_2 \triangleq j' = j + 1 \\ \quad \vee j' = j \end{array} \right) \rightarrow \left(\begin{array}{l} V \triangleq i, j \in \mathbb{N} \\ l \triangleq i = 0 \wedge j = 0 \\ R \triangleq i' = i + 1 \wedge j' = j \\ \quad \vee i' = i \wedge j' = j + 1 \\ \quad \vee i' = i + 1 \wedge j' = j + 1 \\ \quad \vee i' = i \wedge j' = j \end{array} \right)$$

