

# Systèmes de transition

Philippe Quéinnec, Xavier Thirioux, Aurélie Hurault

ENSEEIH

Département Sciences du Numérique

## Contexte

- Système critique, dont la défaillance entraîne des conséquences graves (exemple : médical, transport)
- Système complexe, dont il est difficile de se convaincre de la correction (exemple : systèmes concurrents)

## Pourquoi ?

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

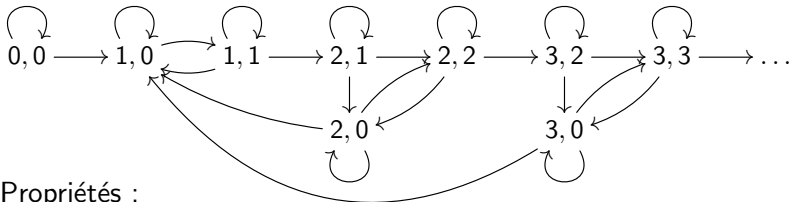
## Comment ?

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

Soit trois processus exécutant concurremment (par entrelacement) :

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

❶ Description du système en termes d'états ?



❷ 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 transition
- 5 Outils : vérificateur automatique, assistant de preuve

Auteur principal : **Leslie Lamport**



# Plan du cours

- 1 (C) Le cadre formel : systèmes de transition
- 2 (CTD) Spécification opérationnelle :  $TLA^+$  – les actions (1)
- 3 (CTD) Spécification opérationnelle :  $TLA^+$  – les actions (2)
- 4 (TP) Recherche de solutions par accessibilité
- 5 (C) Contrôler la progression : l'équité
- 6 (C) Énoncer des propriétés : la logique temporelle linéaire LTL
- 7 (CTD) Logique temporelle et équité dans  $TLA^+$
- 8 (CTD) Étude d'un algorithme d'exclusion mutuelle
- 9 (CTD,TP) Étude d'un algorithme distribué
- 10 (C) Énoncer des propriétés : logique temporelle arborescente CTL
- 11 (TP) Concurrence : allocation de ressources
- 12 (C) Vérification par preuve et vérification de modèles
- 13 (TP) Étude d'un protocole de communication en mémoire partagée

## Ressources

- moodle : supports de cours, TP, examens
- <http://lamport.azurewebsites.net/video/videos.html>  
vidéos de L. Lamport sur TLA<sup>+</sup>
- <http://lamport.azurewebsites.net/tla/tla.html>  
autres ressources (livre *Specifying Systems*)
- <https://learntla.com/>  
guide d'introduction à TLA<sup>+</sup> (exemples surtout en PlusCal)



# Première partie

## Systèmes de transition



# Introduction

## Objectifs

Représenter les exécutions d'un algorithme 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 transition peut être construit :

- *avant* l'écriture du programme, pour explorer la faisabilité de l'algorithme.  
Le programme final est un raffinement en utilisant le système de transition comme guide.
- *après* l'écriture du programme, par abstraction, en ne conservant que les aspects significatifs du programme concret pour obtenir le principe de l'algorithme.

Plutôt que prouver des programmes concrets, on prouve des algorithmes.



# Plan

- 1 Définitions
  - Système de transition
  - Traces, exécutions
  - États, graphe
  - Système de transition étiqueté
- 2 Représentations
  - Explicite
  - Implicite
- 3 Propriétés générales
  - Blocage
  - Réinitialisable
  - Bégaiement
- 4 Composition de systèmes de transition



# Système de transition



## Système de transition (ST)

Un système de transition 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  $s'$ .

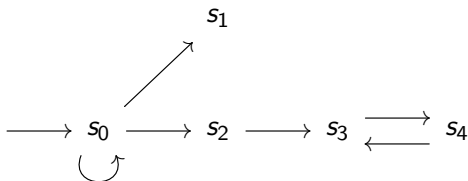


## Exemple - système de transition

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

$$I = \{s_0\}$$

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



# 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  $\sigma$ .

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  $\sigma$  :

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



# Traces finies



## Traces finies

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

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 transition.

Une trace finie  $\langle s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_{n-1} \rightarrow s_n \rangle \in S^*$  est **maximale**

$\triangleq$  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 et traces issues d'un état



### Traces infinies

Soit  $\langle S, I, R \rangle$  un système de transition, 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 issues d'un état

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

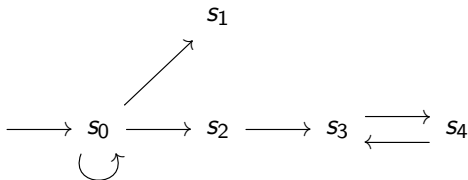
$Exec(\mathcal{S}) \triangleq$  l'ensemble des exécutions de  $\mathcal{S} = \bigcup_{s_0 \in I} Traces(s_0)$ .

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





## Exemple - traces, exécutions



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

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

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

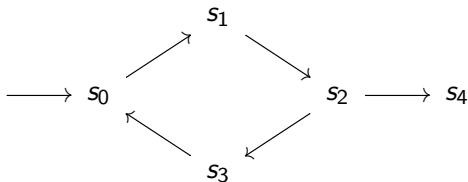
$$\text{Traces}(s_2) = \langle s_2 \rightarrow (s_3 \rightarrow s_4)^\omega \rangle$$

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

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



## Exemple 2 - traces, exécutions

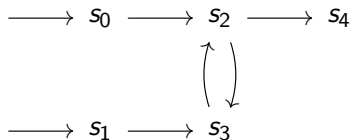


$$\text{Traces}(s_2) =$$

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

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

## 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 transition.

$s \in S$  est un état **accessible**  $\triangleq$  il existe une exécution qui passe par  $s$  (ou équivalent, il existe un préfixe d'exécution qui aboutit à  $s$ ).

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



## Graphe des exécutions

### Graphe des exécutions

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

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 transition étiqueté



## ST étiqueté

Un système de transition 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 semble se rapprocher des automates.

Mais : exécutions infinies, pas d'état terminal, pas de contrôle externe des transitions.



# Équivalence aux ST sans étiquette



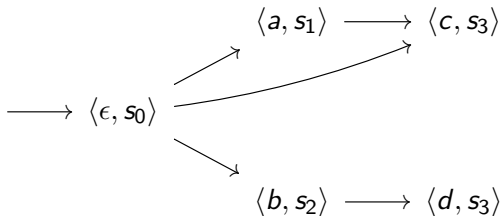
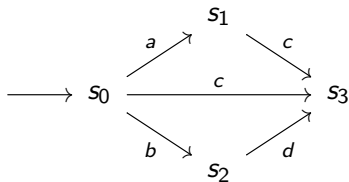
Un système de transition étiqueté  $\langle S, I, R, L, Etiq \rangle$  est équivalent au système sans étiquette  $\langle S', I', R' \rangle$  défini par :

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

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 entre système de transition et automate

### Système de transition $\neq$ automate à états finis

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

Contrairement aux automates à états finis, les systèmes de transition sont Turing-complets, i.e. permettent de décrire n'importe quel calcul.



# Plan

- 1 Définitions
  - Système de transition
  - Traces, exécutions
  - États, graphe
  - Système de transition étiqueté
- 2 Représentations
  - Explicite
  - Implicite
- 3 Propriétés générales
  - Blocage
  - Réinitialisable
  - Bégaiement
- 4 Composition de systèmes de transition



# Représentation en extension



## 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 transition où le nombre d'états et de transitions est fini.



## Représentation en intention



Représentation symbolique à l'aide de variables.

### Système de transition à base de variables

Un triplet  $\langle V, Init, Trans \rangle$  où

- $V = \{v_1, \dots, v_n\}$  : ensemble fini de variables.
- $Init(v_1, \dots, v_n)$  : prédicat définissant les états initiaux et portant sur les variables  $v_i$ .
- $Trans(v_1, \dots, v_n, v'_1, \dots, v'_n)$  : 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;
}
```

En extension pour  $N = 5$  :

$\langle (0, 1, 2, 3, 4, 5), \{0\}, \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)\} \rangle$

Graphe d'exécution pour  $N = 5$  :

————→ 0 —————→ 1 —————→ 2 —————→ 3 —————→ 4 —————→ 5

Symboliquement (en intention) :

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

$$I \triangleq i = 0$$

$$T \triangleq i < N \wedge i' = i + 1 \quad \text{ou} \quad T \triangleq i' \leq N \wedge i' - i = 1$$



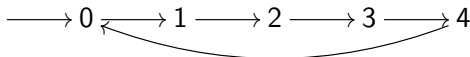
## Exemple : un compteur cyclique

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

En extension pour  $N = 5$  :

$\langle (0, 1, 2, 3, 4, 5), \{0\}, \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 0)\} \rangle$

Graphe d'exécution pour  $N = 5$ .



Symboliquement :

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

$$I \triangleq i = 0$$

$$T \triangleq i' = (i + 1) \bmod N$$

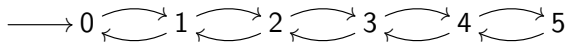
## Exemple : un entier oscillant



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

En extension pour  $N = 5$  :  $\langle (0, 1, 2, 3, 4, 5), \{0\}, \{(0, 1), (1, 0), (1, 2), (2, 1), (2, 3), (3, 2), (3, 4), (4, 3), (4, 5), (5, 4)\} \rangle$

Graphe d'exécution pour  $N = 5$ .



Symboliquement :

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

$$I \triangleq i = 0$$

$$T \triangleq i > 0 \wedge i' = i - 1 \quad \text{ou} \quad T \triangleq |i' - i| = 1 \wedge 0 \leq i' \leq N \\ \vee i < N \wedge i' = i + 1$$

## Système de transition correspondant



Pour une description symbolique  $\langle V, Init, Trans \rangle$ , le système de transition correspondant est  $\langle S, I, R \rangle$  où :

- $S = \prod_{i \in 1..n} D_i$ ,  
où  $D_1, \dots, D_n$  sont les domaines (types) des variables  $v_1, \dots, v_n$
- $I = \{(v_1, \dots, v_n) \mid Init(v_1, \dots, v_n)\}$
- $R = \{((v_1, \dots, v_n), (v'_1, \dots, v'_n)) \mid Trans(v_1, \dots, v_n, v'_1, \dots, v'_n)\}$





# 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 transition est un prédicat portant sur les variables (d'état) primées et non primées.

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



## Exemple - prédicats

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

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

$$T \triangleq n \neq 1 \wedge \left( \begin{array}{l} (n' = n/2 \wedge n \equiv 0[2]) \\ \vee (n' = (3n+1)/2 \wedge n \equiv 1[2]) \end{array} \right)$$

Prédicats d'état :  $I, n < 20$

Prédicats de transition :  $T, n' - n > 3$



# Plan

- 1 Définitions
  - Système de transition
  - Traces, exécutions
  - États, graphe
  - Système de transition étiqueté
- 2 Représentations
  - Explicite
  - Implicite
- 3 Propriétés générales
  - Blocage
  - Réinitialisable
  - Bégaiement
- 4 Composition de systèmes de transition



# 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.



## 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 de transitions pour revenir à l'état initial du système et ainsi redémarrer. Un tel système n'a que des exécutions infinies.



# Bégaiement

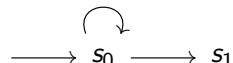


## Bégaiement

Un état  $s$  bégaie  $\triangleq$  l'état possède une boucle :  $(s, s) \in R$ .

Un système de transition bégaie  $\triangleq$  tout état possède une boucle vers lui-même :  $Id \subseteq R$ .

Utilité :

- 

Modéliser l'avancement arbitraire :  $\longrightarrow s_0 \longrightarrow s_1$   
*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 bégaiement) a un unique successeur avec bégaiement : lui-même. La terminaison (l'interblocage)  $\dots \rightarrow s_i$  est alors  $\dots \rightarrow s_i^\omega$ .
- Composer plusieurs systèmes de transition.



# Plan

- 1 Définitions
  - Système de transition
  - Traces, exécutions
  - États, graphe
  - Système de transition étiqueté
- 2 Représentations
  - Explicite
  - Implicite
- 3 Propriétés générales
  - Blocage
  - Réinitialisable
  - Bégaiement
- 4 Composition de systèmes de transition



## Composition : produit libre



### Produit libre

La composition des ST avec bégaiement  $\langle V_1, I_1, T_1 \rangle$  et  $\langle V_2, I_2, T_2 \rangle$  est  $\langle V, I, T \rangle$  où :

- $V \stackrel{\Delta}{=} V_1 \cup V_2$  (union des variables)
- $I \stackrel{\Delta}{=} I_1 \wedge I_2$  (chaque sous-système démarre dans un de ses états initiaux)
- $T \stackrel{\Delta}{=} T_1 \wedge T_2$  (chaque sous-système évolue selon ses transitions)

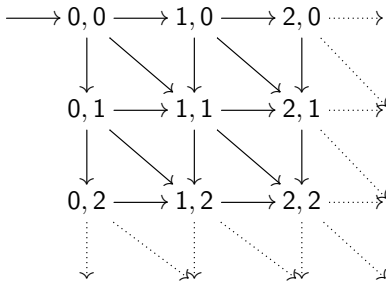
Comme  $T_1$  et  $T_2$  peuvent bégayer,  $T_1 \wedge T_2$  signifie donc qu'on peut exécuter une transition de  $T_1$  seule et  $T_2$  bégayant, ou bien réciproquement, ou bien encore exécuter  $T_1$  en même temps que  $T_2$ .





## Exemple - produit libre

$$\left( \begin{array}{l} V_1 \triangleq i \in \mathbb{N} \\ l_1 \triangleq i = 0 \\ T_1 \triangleq i' = i + 1 \\ \vee i' = i \end{array} \right) \otimes \left( \begin{array}{l} V_2 \triangleq j \in \mathbb{N} \\ l_2 \triangleq j = 0 \\ T_2 \triangleq j' = j + 1 \\ \vee j' = j \end{array} \right) \rightarrow \left( \begin{array}{l} V \triangleq i, j \in \mathbb{N} \\ l \triangleq i = 0 \wedge j = 0 \\ T \triangleq i' = i + 1 \wedge j' = j \\ \vee i' = i \wedge j' = j + 1 \\ \vee i' = i + 1 \wedge j' = j + 1 \\ \vee i' = i \wedge j' = j \end{array} \right)$$



+bégaiement

## Composition : produit synchronisé strict (ou fermé)



### Produit synchronisé strict

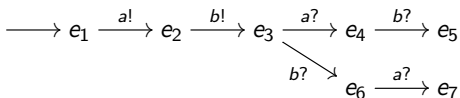
Le produit synchronisé des ST étiquetés  $\langle S_1, I_1, R_1, L_1 \rangle$  et  $\langle S_2, I_2, R_2, L_2 \rangle$  est  $\langle S, I, R, L \rangle$  où :

- $S \stackrel{\Delta}{=} S_1 \times S_2$  (couple d'états)
- $I \stackrel{\Delta}{=} I_1 \times I_2$   
(chaque sous-système démarre dans un de ses états initiaux)
- $R \stackrel{\Delta}{=} \{((s_1, s_2), (s'_1, s'_2)) \mid (s_1, s'_1) \in R_1 \wedge (s_2, s'_2) \in R_2 \wedge \text{Etiqu}((s_1, s'_1)) = \text{Etiqu}((s_2, s'_2))\}$   
(les deux sous-systèmes évoluent selon des transitions portant les mêmes étiquettes)
- $L = L_1 \cap L_2$  (étiquettes communes seulement)

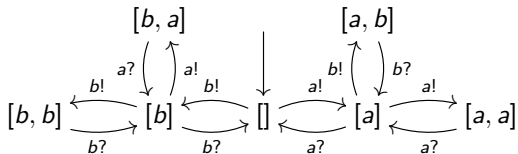
## Exemple – produit synchronisé strict



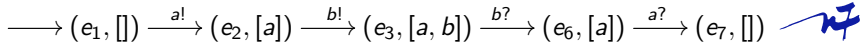
Interprétation :  $a$  et  $b$  sont des messages,  $a!$  /  $a?$  sont l'envoi / la réception d'un message. Le premier ST est une application quelconque et le deuxième décrit les propriétés de la communication.



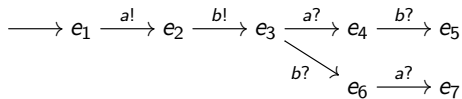
Synchronisé strict avec LIFO à 2 éléments (pile)



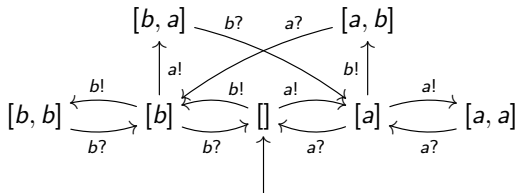
Donne



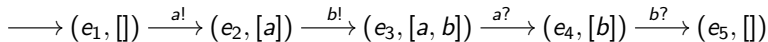
# Exemple – produit synchronisé strict



Synchronisé strict avec FIFO à 2 éléments (file)



Donne



## Composition : produit synchronisé ouvert



### Produit synchronisé ouvert

Le produit synchronisé des ST étiquetés  $\langle S_1, I_1, R_1, L_1 \rangle$  et  $\langle S_2, I_2, R_2, L_2 \rangle$  est  $\langle S, I, R, L \rangle$  où :

- $S \triangleq S_1 \times S_2$  (couple d'états)
- $I \triangleq I_1 \times I_2$
- $R \triangleq \left\{ \begin{array}{l} ((s_1, s_2), (s'_1, s'_2)) \mid (s_1, s'_1) \in R_1 \wedge (s_2, s'_2) \in R_2 \\ \quad \wedge \text{Etiqu}((s_1, s'_1)) = \text{Etiqu}((s_2, s'_2)) \\ ((s_1, s_2), (s'_1, s_2)) \mid (s_1, s'_1) \in R_1 \wedge \text{Etiqu}((s_1, s'_1)) \notin L_2 \\ ((s_1, s_2), (s_1, s'_2)) \mid (s_2, s'_2) \in R_2 \wedge \text{Etiqu}((s_2, s'_2)) \notin L_1 \end{array} \right\}$
- $L = L_1 \cup L_2$

Synchronisation sur étiquette commune, bégaiement sur étiquette absente.

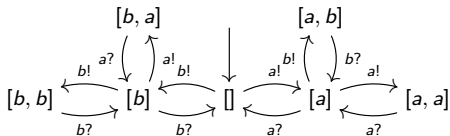


# Exemple – produit synchronisé ouvert



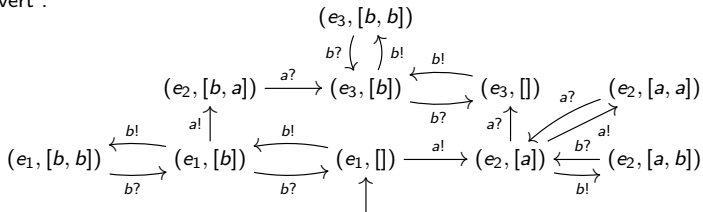
$$\longrightarrow e_1 \xrightarrow{a!} e_2 \xrightarrow{a?} e_3$$

Synchronisé avec LIFO à 2 éléments (pile)



Donne

- strict :  $\longrightarrow (e_1, []) \xrightarrow{a!} (e_2, [a]) \xrightarrow{a?} (e_3, [])$
- ouvert :



# Bilan

Cette séance a présenté :

- la définition de système de transition (états, transitions)
- la notion de trace et d'exécution
- la représentation explicite (en extension) ou symbolique (en intention)
- quelques propriétés génériques, dont le bégaiement
- diverses formes de composition de systèmes de transition

