Deuxième partie

# CCS
# Calculus of Communicating Systems

# Plan

# Appplication : CCS

- CCS = Calculus of Communicating Systems.
- Calcul (algèbre) de processus.
- Créé en 1988 par Robin Milner.
- Axé sur la représentation des processus et des communications.
- Aucune structure de données.
- Permet la description de systèmes concurrents, distribués, à objets, à composants, etc.

# Syntaxe CCS

## Définition 1 (Alphabet CCS)

- Le langage d'événements contient des paires d'événements complémentaires appelés *actions*.

$$L \triangleq \{\tau\} \cup \mathcal{A} \cup \overline{\mathcal{A}}$$

$\mathcal{A}$ est l'ensemble des actions
$\overline{\mathcal{A}} \triangleq \{\overline{a} \mid a \in \mathcal{A}\}$ l'ensemble des actions complémentaires.

- $\mathcal{A}$ correspond aux événements de réception de message.
- $\overline{\mathcal{A}}$ correspond aux événements d'émission de message.

# Syntaxe CCS

## Définition 2 (Termes CCS)

Les termes CCS sont décrits par la grammaire suivante :

| | | | |
|---|---|---|---|
| $\mathcal{P}$ | $::=$ | $0$ | processus terminé |
| | $\mid$ | $l.\mathcal{P}$ | construction préfixe |
| | | | On exécute l'événement $l \in L$, puis $\mathcal{P}$ |
| | $\mid$ | $\mathcal{P}\|\|\mathcal{P}$ | construction parallèle |
| | | | les 2 $\mathcal{P}$ s'exécutent comme des threads |
| | $\mid$ | $\mathcal{P} + \mathcal{P}$ | construction choix |
| | | | un seul des $\mathcal{P}$ s'exécute |
| | $\mid$ | $\nu a.\mathcal{P}$ | déclaration d'un canal local à $\mathcal{P}$ |

# Syntaxe CCS

## Définition 3 (Processus CCS)

- À cette grammaire s'ajoutent des définitions de processus (éventuellement récursives), sous forme de système d'équations :

$$\{X_i \triangleq E_i\}$$

où les $E_i$ sont des termes CCS faisant intervenir les variables de processus $X_i$.

- Ces définitions doivent être **protégées**, i.e. les variables $X_i$ apparaissant derrière une étiquette préfixe.
- Cette contrainte permet l'existence d'une solution unique.
- Un processus CCS est alors un des $X_i$.

# Syntaxe CCS

## Définition 4 (Programme CCS)

Un programme CCS $\mathcal{G}$ est donc un ensemble de définitions de processus $\mathcal{P}$

$$
\begin{array}{lcl}
\mathcal{G} & ::= & \mathcal{D} \mid \mathcal{D}\,\mathcal{G} \\
\mathcal{D} & ::= & X \triangleq \mathcal{P} \\
\mathcal{P} & ::= & 0 \mid I.\mathcal{P}' \mid \mathcal{P}||\mathcal{P} \mid \mathcal{P}+\mathcal{P} \mid \nu \mathrm{a}.\mathcal{P} \\
\mathcal{P}' & ::= & X \mid 0 \mid I.\mathcal{P}' \mid \mathcal{P}'||\mathcal{P}' \mid \mathcal{P}'+\mathcal{P}' \mid \nu \mathrm{a}.\mathcal{P}'
\end{array}
$$

# Définitions protégées

- Définitions correctes :

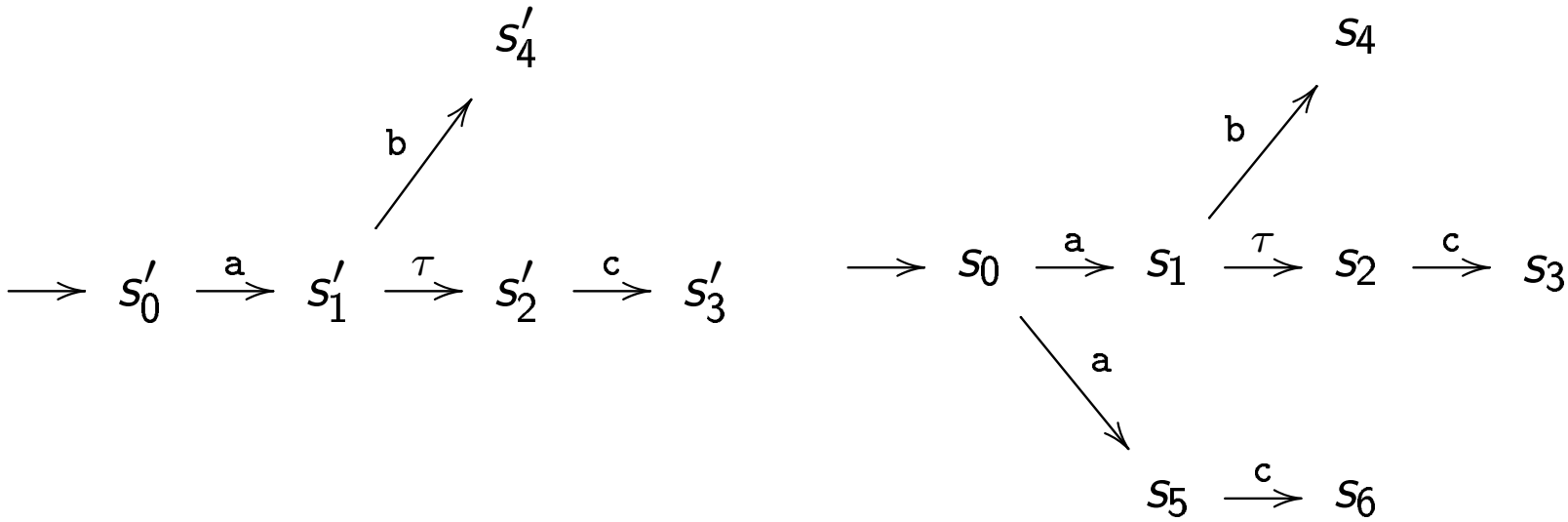
$$X \triangleq \mathrm{a}.X \qquad X \triangleq \mathrm{a}.0||\mathrm{b}.X \qquad X \triangleq \mathrm{a}.(\mathrm{b}.0||X)$$

- Définitions incorrectes :

$$X \triangleq X \qquad X \triangleq \mathrm{a}.0 + (\mathrm{b}.0||X)$$

# Exemple 1

Soient $\mathcal{S}$ et $\mathcal{S}'$ 2 S.T.E. définis sur le même alphabet :




On peut les représenter par les 2 processus CCS suivants :

$P' \triangleq \mathrm{a}.(\mathrm{b}.0 + \tau.\mathrm{c}.0)$ $\qquad$ $P \triangleq \mathrm{a}.(\mathrm{b}.0 + \tau.\mathrm{c}.0) + \mathrm{a}.\mathrm{c}.0$

# Modélisation

- Modélisons un distributeur de boissons. Pour 20 centimes, on peut avoir un café ou un thé, et pour 40 centimes, on peut avoir un cappuccino. L'appareil ne rend pas la monnaie et ne reconnaît que les pièces de 10 et 20 centimes :

$$\begin{aligned} \mathit{Machine} &\triangleq 20.\mathit{Service}_1 + 10.10.\mathit{Service}_1 \\ \mathit{Service}_1 &\triangleq \overline{\texttt{cafe}}.\mathit{Machine} + \overline{\texttt{the}}.\mathit{Machine} \\ &\quad +20.\mathit{Service}_2 + 10.10.\mathit{Service}_2 \\ \mathit{Service}_2 &\triangleq \overline{\texttt{cappuccino}}.\mathit{Machine} \end{aligned}$$

- De même, un utilisateur qui souhaite et peut s'offrir un café avec des pièces de 10 centimes peut être modélisé par le terme CCS :

$$\mathit{Utilisateur} \triangleq \overline{10}.\overline{10}.\texttt{cafe}.0$$

- Il reste à déterminer comment cet utilisateur et cette machine peuvent interagir.

# Sémantique

- On définit une sémantique opérationnelle pour CCS, i.e. on définit un S.T.E. pour chaque processus CCS.
- Pour simplifier la description des transitions, on identifie certains termes "égaux" (par ex., $A + B \equiv B + A$).
- Cette équivalence est en fait une congruence % aux opérateurs CCS.
- **Donc :** sémantique = relation de congruence + relation de transition.

# Congruence

- Une congruence, par rapport à une algèbre donnée, est une équivalence qui commute avec les opérateurs de cette algèbre.
- Par ex, la congruence modulo $K$ pour les opérateurs de $\mathbb{Z}$.

## Définition 5 (Notion de congruence CCS)

Une équivalence $\cong$ entre processus CCS est une congruence si et seulement si, pour tous processus tels que $P \cong Q$, on a :

- $\mathrm{a}.P \cong \mathrm{a}.Q$
- $P + M \cong Q + M$ et $M + P \cong M + Q$
- $P||M \cong Q||M$ et $M||P \cong M||Q$
- $\nu\mathrm{a}.P \cong \nu\mathrm{a}.Q$

# Congruence

## Définition 6 (Congruence structurelle)

La congruence structurelle $P \equiv Q$ est définie par les règles suivantes. Elle vérifie les conditions de la définition 5.

- Changement de nom par $\alpha$-conversion : si b n'apparaît pas dans $P$, alors $\nu \mathrm{a}.P \equiv \nu \mathrm{b}.P[\mathrm{a} \leftarrow \mathrm{b}]$.
- $+$ est associative et commutative : $P + Q \equiv Q + P$ et $P + (Q + R) \equiv (P + Q) + R \equiv P + Q + R$.
- $(0, ||)$ est un monoïde commutatif : $P||0 \equiv P$, $P||Q \equiv Q||P$ et $P||(Q||R) \equiv (P||Q)||R \equiv P||Q||R$.
- $\nu \mathrm{a}.0 \equiv 0$, $\nu \mathrm{a}.\nu \mathrm{b}.P \equiv \nu \mathrm{b}.\nu \mathrm{a}.P$.
- Extrusion de portée : si a n'apparaît pas dans $Q$, alors $\nu \mathrm{a}.(P||Q) \equiv (\nu \mathrm{a}.P)||Q$.
- Dépliage des définitions : si $Q \triangleq def$, alors $P \equiv P[Q \leftarrow def]$.

# Propriétés de la congruence structurelle

- Si $P \equiv Q$, alors $P$ et $Q$ admettent le même graphe de transitions.
- En particulier : $P \equiv Q \Rightarrow P \sim Q$.
- Existence d'une forme normale.

**Définition 7 (Forme normale)**

Tout processus $P$ est congruent à un terme sous forme normale, i.e. de la forme : $\nu \vec{a}.(M_1 || \ldots || M_n)$ où les $M_i$ sont des sommes non vides. De plus, on peut ordonner les $M_i$ afin d'obtenir une forme normale unique.

Pour obtenir la forme normale, il suffit d'utiliser l'$\alpha$-conversion et l'extrusion.

# Transitions

## Définition 8 (Relation de transition CCS)

La relation de transition est définie à partir des règles suivantes :

$$\text{Sum}\ \frac{P \xrightarrow{l} P'}{P + Q \xrightarrow{l} P'} \qquad \text{Comm}\ \frac{P \xrightarrow{l} P' \quad Q \xrightarrow{\bar{l}} Q'}{P||Q \xrightarrow{\tau} P'||Q'}$$

$$\text{New}\ \frac{P \xrightarrow{l} P' \quad l \neq \mathrm{a}, \bar{\mathrm{a}}}{\nu \mathrm{a}.P \xrightarrow{l} \nu \mathrm{a}.P'} \qquad \text{Act}\ \frac{}{l.P \xrightarrow{l} P}$$

$$\text{Par}\ \frac{P \xrightarrow{l} P'}{P||Q \xrightarrow{l} P'||Q} \qquad \text{Cong}\ \frac{P \equiv P' \quad P' \xrightarrow{l} Q' \quad Q' \equiv Q}{P \xrightarrow{l} Q}$$

# Transitions

Muni de la relation de congruence et de la relation de transition, on définit le système de transitions représentant les exécutions d'un processus CCS.

## Définition 9 (Système de transitions d'un processus CCS)

Tout terme CCS $P$ définit un système de transitions étiqueté $\langle S, L, I, R \rangle$ tel que :

- $S \triangleq \{[Q]_{\equiv} \mid P \rightarrow^{\star} Q\}$.
- $L \triangleq \{\mathrm{a} \mid \exists s, s' \in S. s \xrightarrow{\mathrm{a}} s'\} \cup \{\tau\}$.
- $I \triangleq \{[P]_{\equiv}\}$.
- $R \triangleq \{\langle P, I, P' \rangle \mid P, P' \in S \ \wedge \ P \xrightarrow{I} P'\}$.

**Note :** On prendra la forme normale de $Q$ comme représentant $[Q]_{\equiv}$.

# Transitions

- Une transition de la forme $P \xrightarrow{a} Q$ représente seulement la potentialité d'un événement a.
- La présence d'un environnement jouant un rôle dual, i.e. proposant des actions complémentaires, est nécessaire.
  En effet :
  - Pour recevoir un message, il faut qu'un processus l'émette.
  - Pour émettre un message, il faut qu'un processus puisse le recevoir.
- On est donc dans un cadre de synchronisation par rendez-vous.
- Les seuls réels pas de calcul sont les communications entre processus.

# Exemple de transition

- Pour alléger la présentation, on ne fait pas figurer le processus terminé final 0.
- Ainsi $\overline{10}.\overline{10}.\texttt{cafe}.0$ se note $\overline{10}.\overline{10}.\texttt{cafe}$
- En pratique, on ne prouve pas chaque transition.

$$\text{Comm}\ \frac{\text{Sum}\ \dfrac{\text{Act}\ \dfrac{}{10.10.Service_1 \xrightarrow{10} 10.Service_1}}{10.10.Service_1 + 20.Service_1 \xrightarrow{10} 10.Service_1} \qquad \text{Act}\ \dfrac{}{\overline{10}.\overline{10}.\texttt{cafe} \xrightarrow{\overline{10}} \overline{10}.\texttt{cafe}}}{\text{Cong}\ \dfrac{10.10.Service_1 + 20.Service_1 || \overline{10}.\overline{10}.\texttt{cafe} \xrightarrow{\tau} 10.Service_1 || \overline{10}.\texttt{cafe}}{Machine || Utilisateur \xrightarrow{\tau} 10.Service_1 || \overline{10}.\texttt{cafe}}}$$

# Expressivité

- Les traces d'un processus CCS ne forment pas un langage régulier dans le cas général.
- L'ensemble des termes CCS est évidemment infini, même à bisimulation prêt.
- CCS (avec définitions paramétrées) est Turing-complet, i.e. toute machine de Turing peut être bisimulée faiblement par un processus CCS, qu'on peut construire.

# Exemple de processus non régulier

Soit le processus $P \triangleq \mathtt{a}.(\mathtt{b}.0||P)$

- Le S.T.E. correspondant à $P$ est :

$$P \underset{\mathtt{b}}{\overset{\mathtt{a}}{\rightleftarrows}} P||\mathtt{b}.0 \underset{\mathtt{b}}{\overset{\mathtt{a}}{\rightleftarrows}} P||\mathtt{b}.0||\mathtt{b}.0 \underset{\mathtt{b}}{\overset{\mathtt{a}}{\rightleftarrows}} \cdots$$

- On peut interpréter ce terme $P$ comme un compteur entier qui est incrémenté avec a et décrémenté avec b.
- Les exécutions permises sont celles pour lesquelles tout préfixe contient plus de a que de b.

# Propriétés liées aux (bi)simulations

- La congruence structurelle est une bisimulation forte, i.e. $P \equiv Q \Rightarrow P \sim Q$.
- En fait, $P \equiv Q$ implique que $P$ et $Q$ correspondent à 2 S.T.E. isomorphes (même graphe sauf les états).
- Réciproquement, toute bisimulation forte est une congruence CCS.
- **Par contre**, en général, une bisimulation faible n'est pas une congruence CCS.
- **Donc** la bisimulation faible n'est pas compositionnelle pour CCS.

# Propriétés liées aux (bi)simulations

Néanmoins, $P \approx P'$ implique :

- $\mathrm{a}.P \approx \mathrm{a}.P'$
- $\tau.P \approx \tau.P'$
- $\mathrm{a}.P + Q \approx \mathrm{a}.P' + Q$
  Attention, ce n'est pas la règle de congruence du $+$ !
- $P||Q \approx P'||Q$ et $Q||P \approx Q||P'$
- $\nu\mathrm{a}.P \approx \nu\mathrm{a}.P'$

Mais par exemple, $\mathrm{a}.0 + \mathrm{b}.0 \not\approx \mathrm{a}.0 + \tau.\mathrm{b}.0$ bien que $\mathrm{b}.0 \approx \tau.\mathrm{b}.0$

**Note :** L'opérateur de composition principal entre processus est $||$

# Propriétés liées aux (bi)simulations

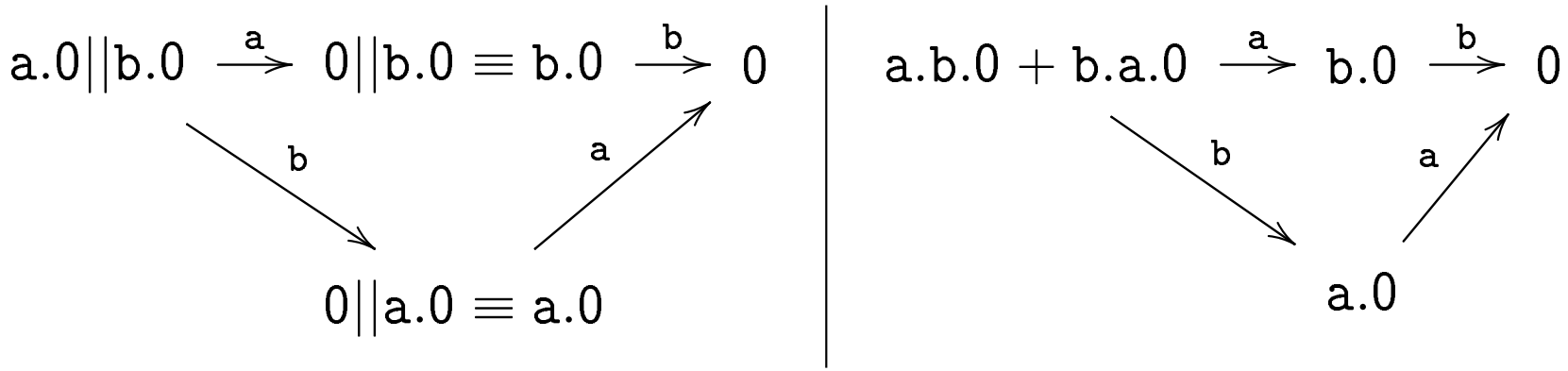
Quelques propriétés utiles (pour tous processus $P$, $Q$ et $R$) :

- $P + P \sim P$
- $P + 0 \sim P$
  Attention au sens de "+"
- $P \sim \sum_{P \xrightarrow{l} Q} l.Q$
- $P \approx \tau.P$
- $P + \tau.P + Q \approx \tau.P + Q$
- $l.P + l.(\tau.P + Q) + R \approx l.(\tau.P + Q) + R$

# Exemples de (bi)simulations

$a.0||b.0 \sim a.b.0 + b.a.0$

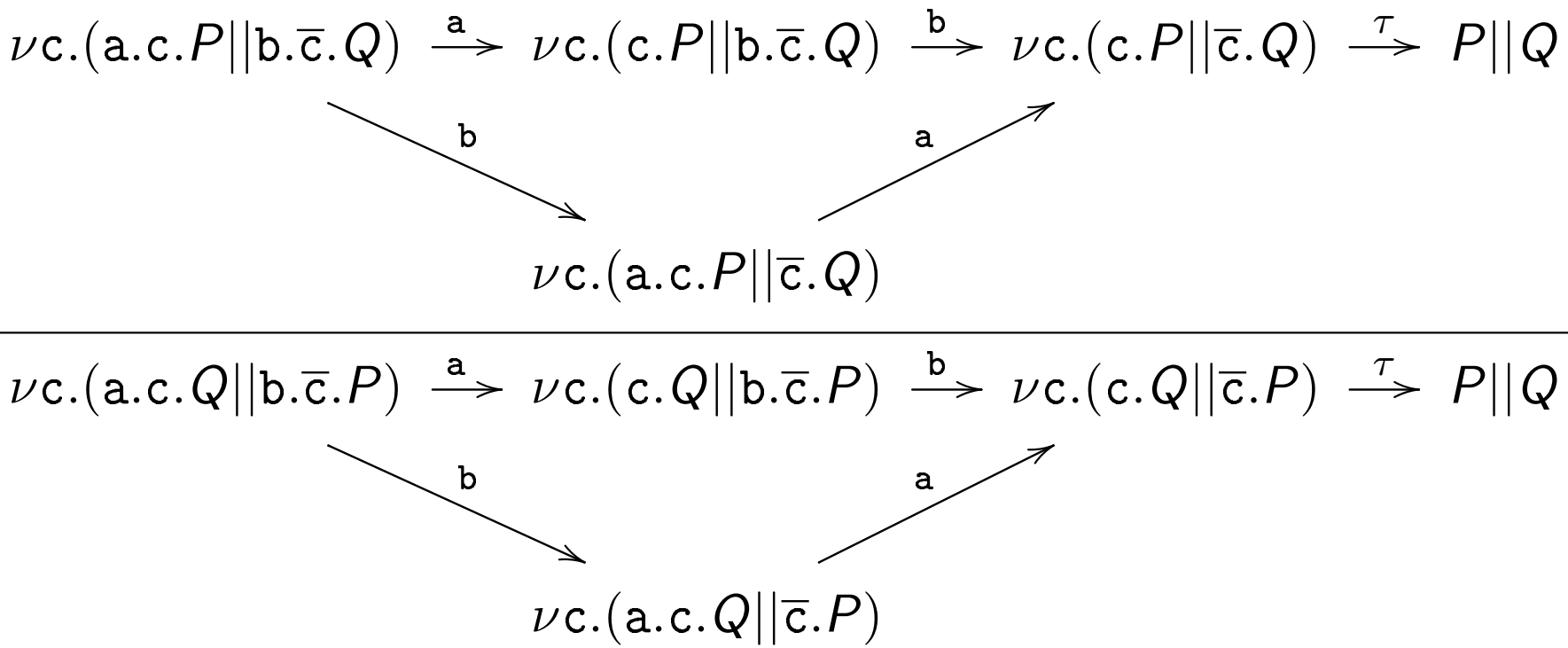
- Les S.T.E. correspondant sont :




- Même systèmes de transitions $\Rightarrow$ fortement bisimilaires.
- Ainsi, deux processus parallèles peuvent être équivalents à une somme.
- On ne peut donc pas connaître la structure interne d'un processus par des tests fondés sur les seules capacités de communication.

# Exemples de (bi)simulations

- $(a.0||b.0) + (a.0||c.0) \prec a.0||(b.0 + c.0)$
- Mais $(a.0||b.0) + (a.0||c.0) \not\sim a.0||(b.0 + c.0)$
- $\nu a.a.P \sim \nu a.\overline{a}.P \sim 0$
- $\nu c.(a.c.P||b.\overline{c}.Q) \sim \nu c.(a.c.Q||b.\overline{c}.P)$




Même S.T.E. $\Rightarrow$ fortement bisimilaires

# Problème de la (bi)simulation en CCS

- Un processus CCS peut correspondre à un S.T.E. infini.
- Les algorithmes de calcul de plus grande (bi)simulation ne s'appliquent pas.
- Ces relations ne sont ni calculables, ni décidables, dans le cas général.
- CCS Turing-complet $\Rightarrow$ indécidabilité de $\approx$.
- **Toutefois**, si l'on se restreint à CCS sans l'opérateur $\nu$, alors la bisimulation forte est décidable.

# Bisimulation forte

L'algorithme est basé sur la **co-induction**.

**Définition 10 (Test de bisimulation forte entre processus CCS sans $\nu$)**

1. Mise en forme normale des processus.
2. Représentation des processus par des vecteurs d'entiers.
3. Construction d'un ordre total (lexicographique) bien fondé entre processus.
4. Construction de la preuve par règles d'inférence.

# Algorithme : Mise en forme normale

Tout système d'équations $\{X_i \triangleq E_i\}$ définissant un ensemble de processus CCS peut se mettre sous la forme :

$$\{X_i \triangleq \sum_{j \in J} \mathrm{a}_j.(X_{j_1} || \ldots || X_{j_n})\}$$

éventuellement en rajoutant des définitions de processus.

# Algorithme : Représentation par vecteurs d'entiers

- L'étape précédente permet de ne s'intéresser qu'à des termes CCS de la forme $X_{i_1} || \ldots || X_{i_n}$.
- À chaque définition $X_i, i \in I$ on associe un entier qui correspond à l'occurrence de $X_i$ dans $X_{i_1} || \ldots || X_{i_n}$.
- Chaque terme parallèle est donc représenté par un unique vecteur $\overrightarrow{v} \in \mathbb{N}^I$.
- On définit enfin un ordre lexicographique (total) entre ces termes/vecteurs.

$$\langle u_1, \ldots, u_n \rangle < \langle v_1, \ldots, v_n \rangle \triangleq u_1 < v_1 \vee (u_1 = v_1 \wedge \langle u_2, \ldots, u_n \rangle < \langle v_2, \ldots, v_n \rangle)$$

- Cet ordre est bien fondé $\Rightarrow$ la preuve termine.

Pour conserver les mêmes conventions que CCS, on notera $||$ la somme de tels vecteurs.

# Algorithme : Règles d'inférence

La preuve de bisimulation tient en l'utilisation des règles suivantes dans l'ordre de préférence Refl, CoInd, Label :

$$\text{Label}\ \frac{\forall P \xrightarrow{l} P'.\exists Q \xrightarrow{l} Q'.P' \sim Q' \qquad \forall Q \xrightarrow{l} Q'.\exists P \xrightarrow{l} P'.P' \sim Q'}{P \sim Q}$$

$$\text{Refl}\ \frac{}{P \sim P}$$

$$\text{CoInd}\ \frac{\overrightarrow{u} < \overrightarrow{v} \qquad \overrightarrow{u}||\overrightarrow{\delta} \sim \overrightarrow{w}}{\dfrac{\overrightarrow{v}||\overrightarrow{\delta} \sim \overrightarrow{w}}{\vdots \qquad\qquad \vdots}\ (\text{Label})}$$

$$\overrightarrow{u} \sim \overrightarrow{v}$$

# Preuve de bisimulation

Soit le système d'équations suivant, pour lequel on veut prouver $X_1 \sim X_2$ :

$$\begin{cases} X_1 & \triangleq & \mathtt{a}.(X_1 || \mathtt{b}.0) \\ X_2 & \triangleq & \mathtt{a}.X_3 \\ X_3 & \triangleq & \mathtt{a}.(X_3 || \mathtt{b}.0) + \mathtt{b}.X_2 \end{cases}$$

# Le $\pi$-calcul

- Inventé également par Milner en 1992.
- Etend CCS par le passage de valeurs lors des communications.
- Introduit la notion de mobilité.
- La règle de communication devient :

$$\text{Comm}\frac{P \xrightarrow{l\langle \widetilde{f_i} \rangle} P' \quad Q \xrightarrow{\overline{l\langle \widetilde{r_i} \rangle}} Q'}{P||Q \xrightarrow{\tau} ([\widetilde{r_i}|\widetilde{f_i}]P')||Q'}$$

- Les valeurs ne sont que des canaux créés par des $\nu$.
- L'extension de la notion de simulation n'est pas triviale, à cause des valeurs transportées.