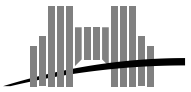
Communication et mobilité

version du 2 janvier 2003 – 20 : 07

Présentation du problème

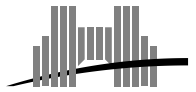


Quoi ?

Les formalismes actuels savent parler seulement de

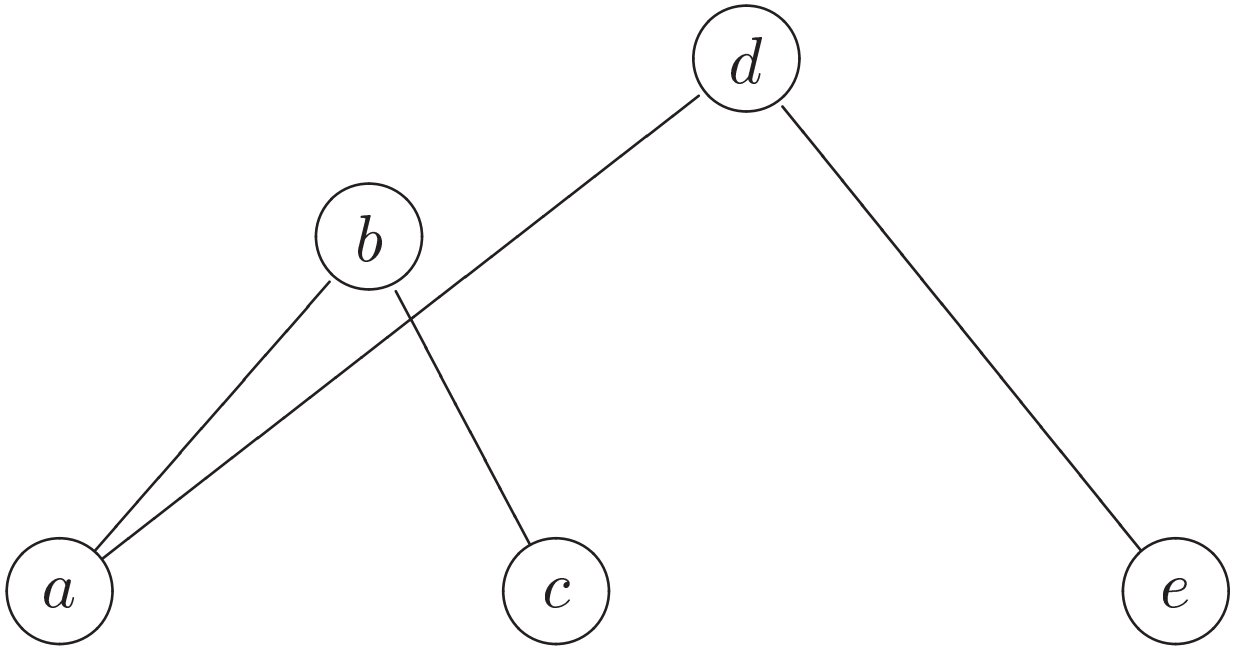
– de modification et d'allocation de la mémoire (machine à registre, etc.)
– d'appel de fonctions (lambda-calcul)

Un des problèmes clés aujourd'hui est celui de la communication et de la mobilité.



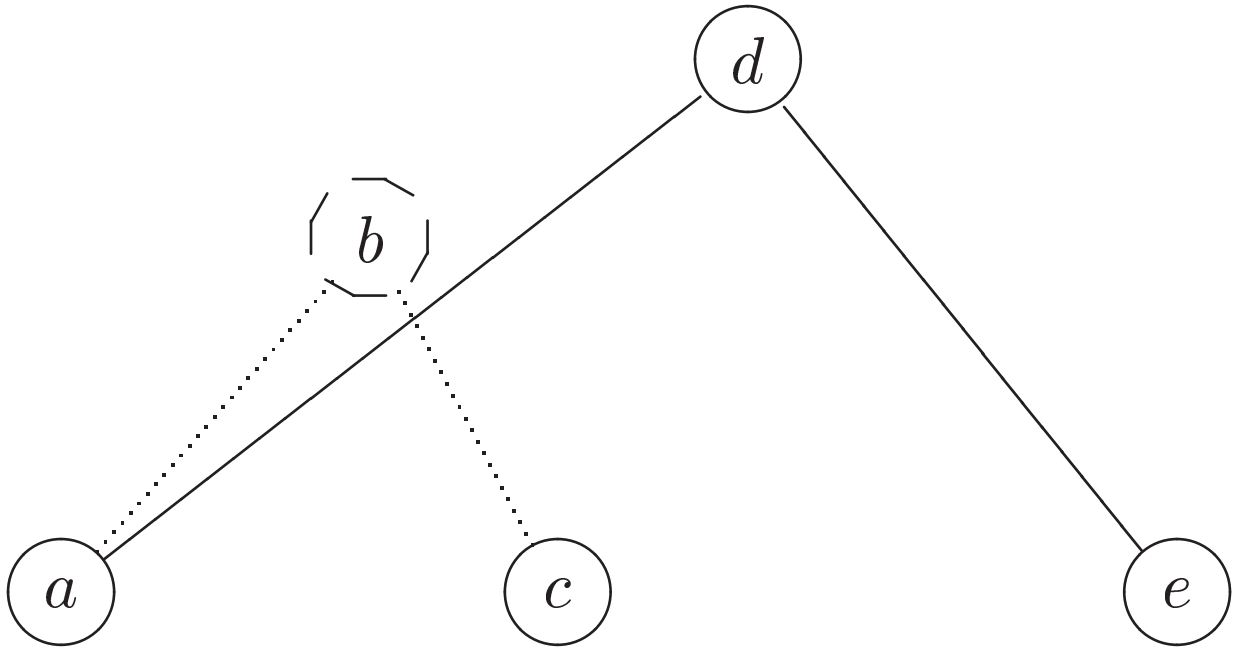
Communication

Il faut décrire les communications.



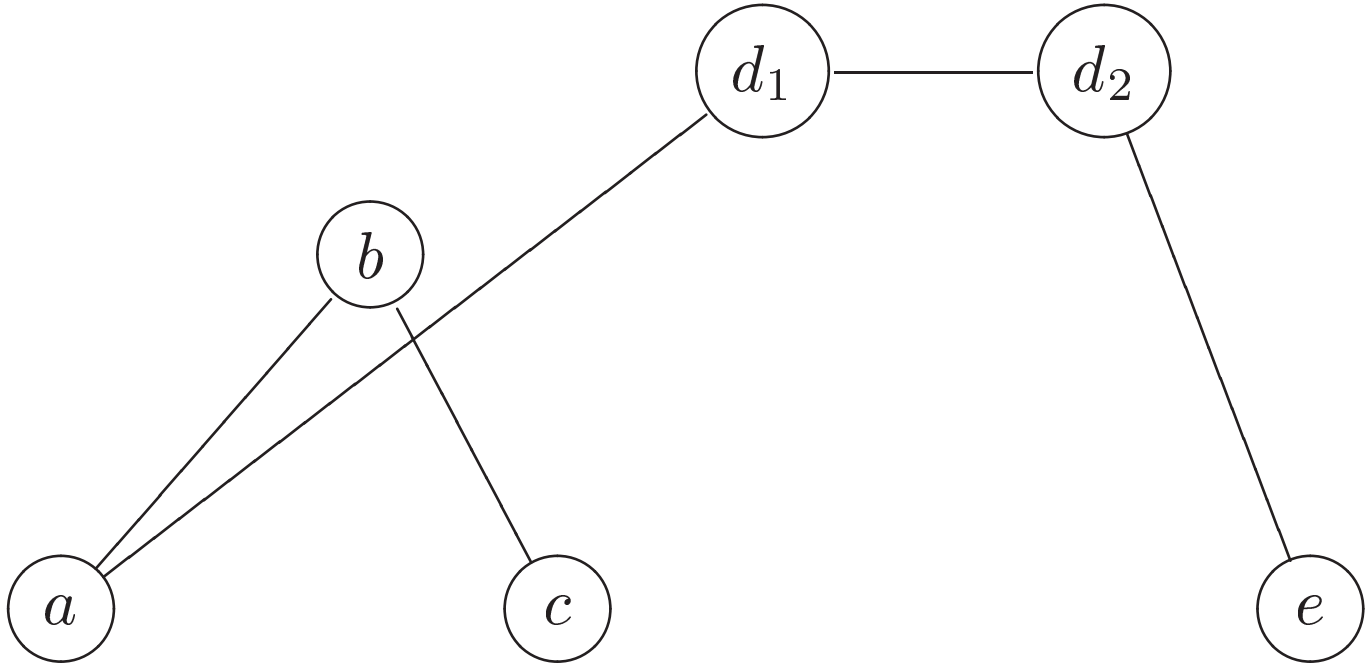
Évolution des communications

On peut supprimer des communications

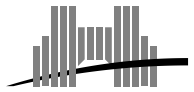


Évolution des communications

Des nœds peuvent se dupliquer



Systèmes de transition étiquetées et bissimulation



Systèmes de transition étiquetées

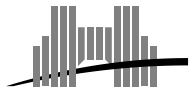
Définition : Un système de transitions étiquetées labelled transition system (LTS) sur Act est une paire $(\mathcal{Q}, \mathcal{T})$ consistant

– d'un ensemble $\mathcal{Q}$ d'états,

– d'une relation ternaire $\mathcal{T} \subseteq \mathcal{Q} \times Act \times \mathcal{Q}$, les transitions.

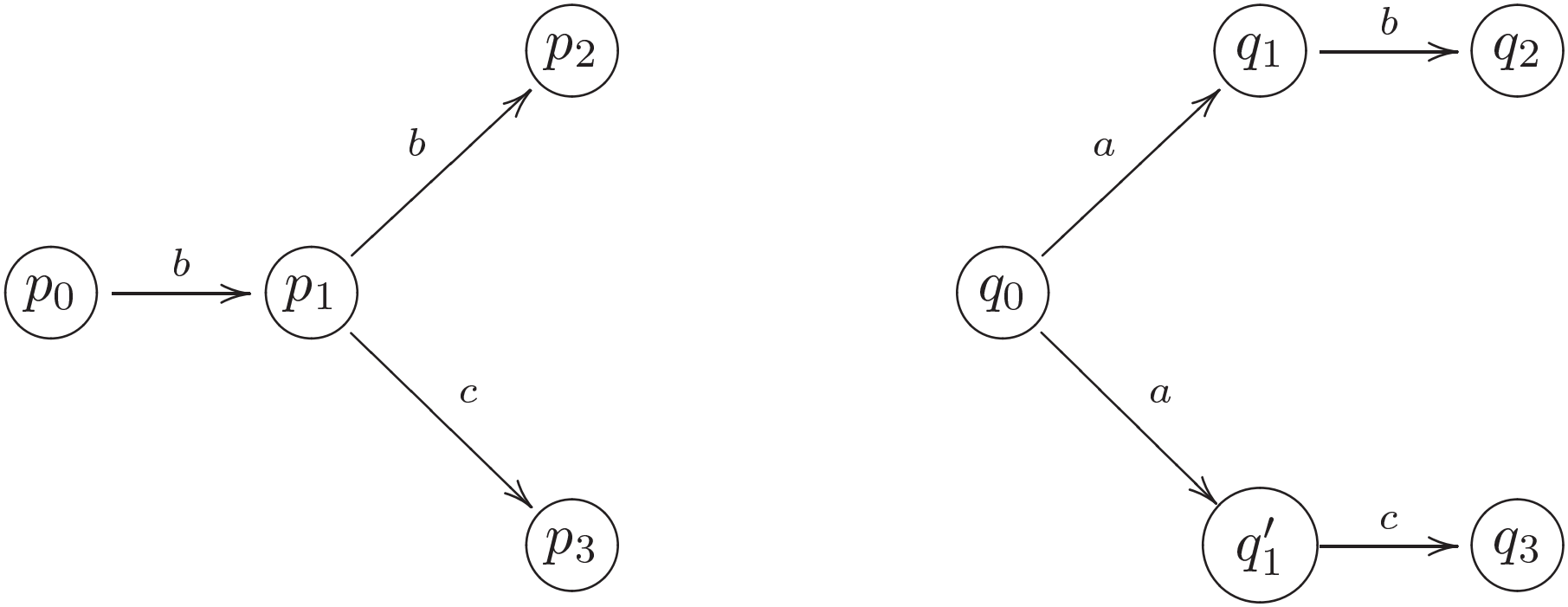
Si $(q, \alpha, q') \in \mathcal{T}$ on écrit $q \xrightarrow{\alpha} q'$ et on dit que q est la source et q' la cible.

Si $q \xrightarrow{\alpha_1} q_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} q_n$, on dit que q_n est dérivé de q par $\alpha_1\alpha_2...\alpha_n$.



Bissimulation

Partant de la source ces deux systèmes ont les mêmes transitions, mais ils sont fondamentalement différents.

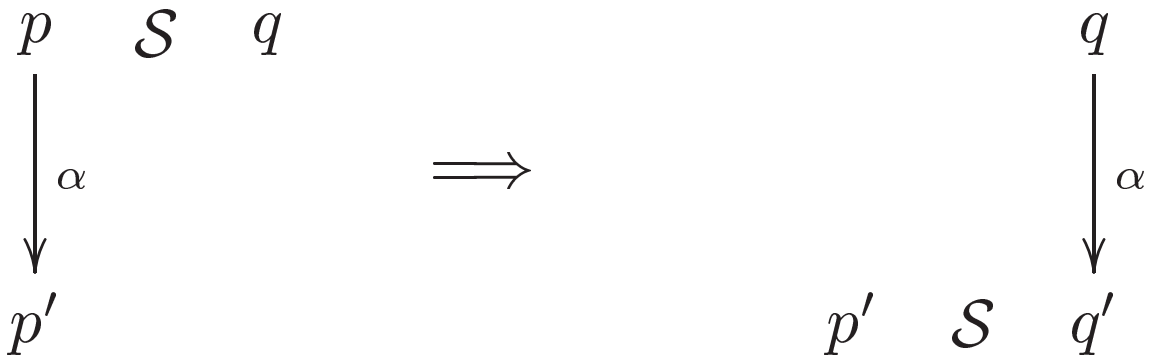


Simulation forte

Définition : Si $\mathcal{S}$ est une relation binaire sur $\mathcal{Q}$ on dit que $\mathcal{S}$ est une simulation forte sur $(\mathcal{Q}, \mathcal{T})$ si

chaque fois que l'on a $p \mathcal{S} q$ et $p \xrightarrow{\alpha} p'$

alors il existe $q \in \mathcal{Q}$ tel que $q \xrightarrow{\alpha} q'$ et $p \mathcal{S} q'$.

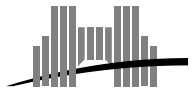


Bissimulation forte

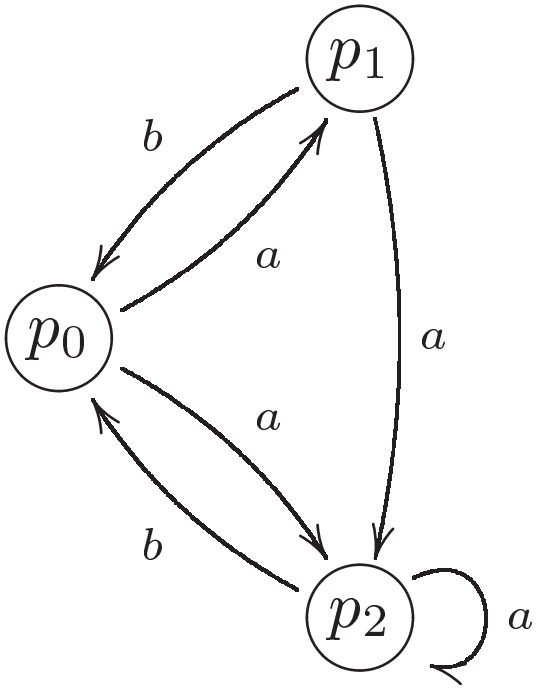
Définition : bissimulation forte Une relation binaire sur $\mathcal{S}$ sur $\mathcal{Q}$ est une bissimulation forte sur $(\mathcal{Q}, \mathcal{T})$

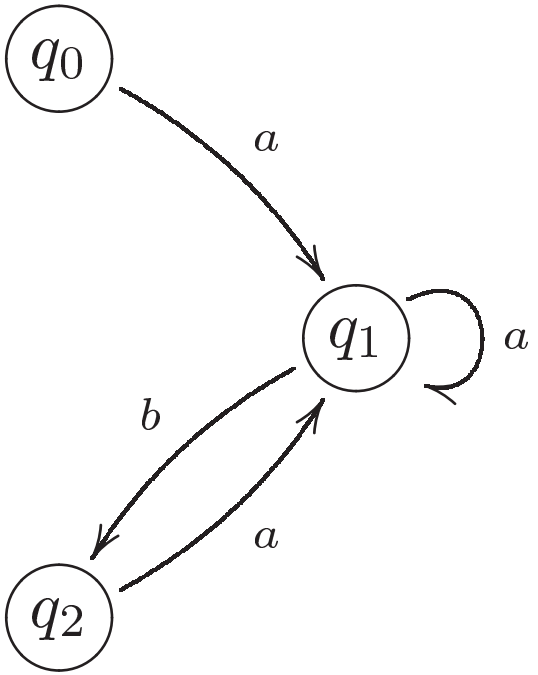
$\mathcal{S}$ est sa converse sont des simulations fortes.

p et p sont fortement bissimilaire, ce qu'on écrit $p \sim q$ s'il existe une bissimulation forte $\mathcal{S}$ telle que $p \, \mathcal{S} \, q$.



Bissimulation, un exemple



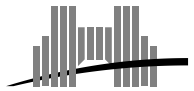


Bissimulation, un exemple

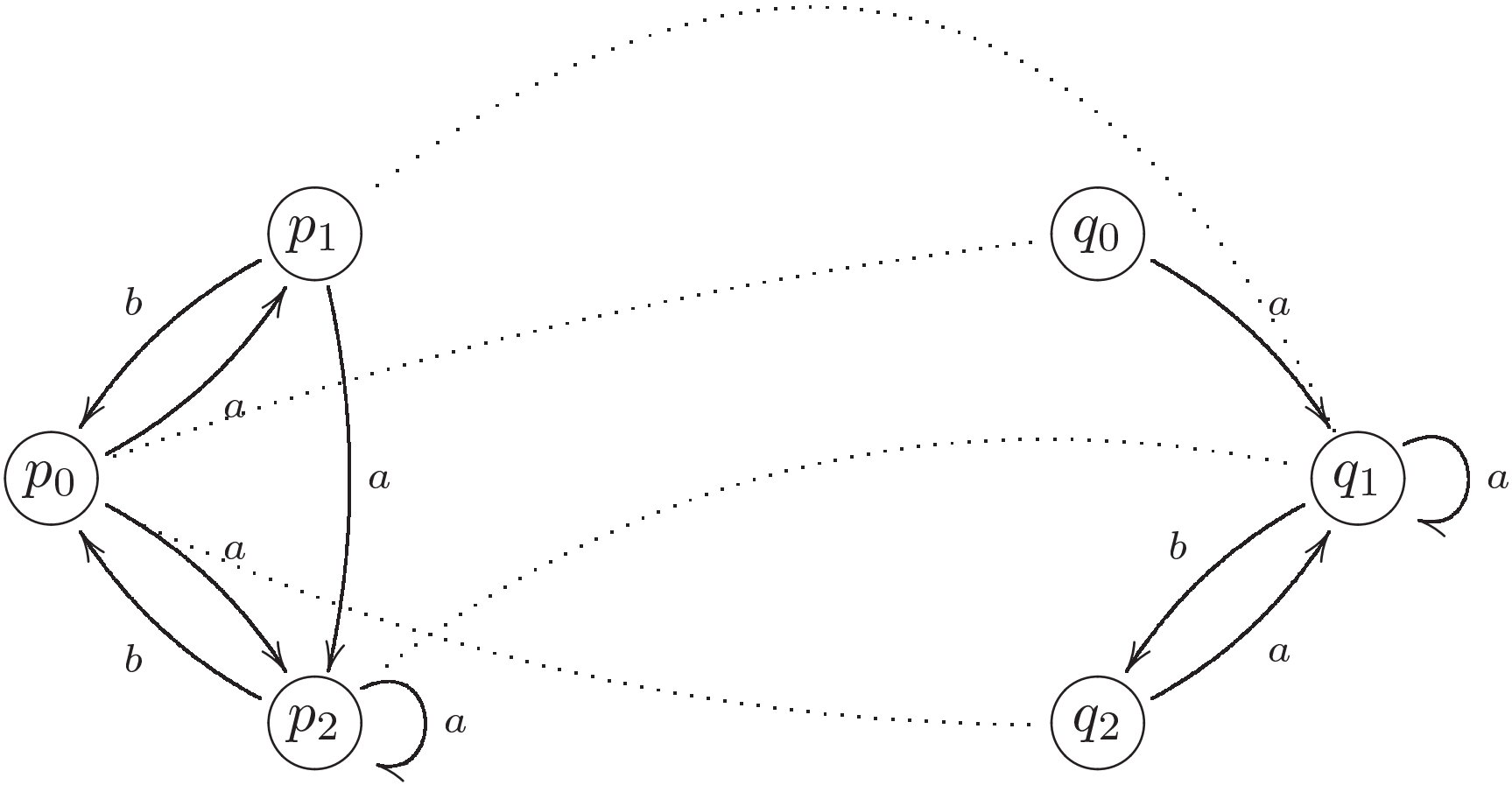
On a $p_0 \sim q_0$.

Pour le prouver, on exhibe

$$\mathcal{S} = \{(p_0, q_0), (p_0, q_2), (p_1, q_1), (p_2, q_1))\}$$

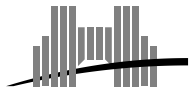


Bissimulation, un exemple



Bissimulation

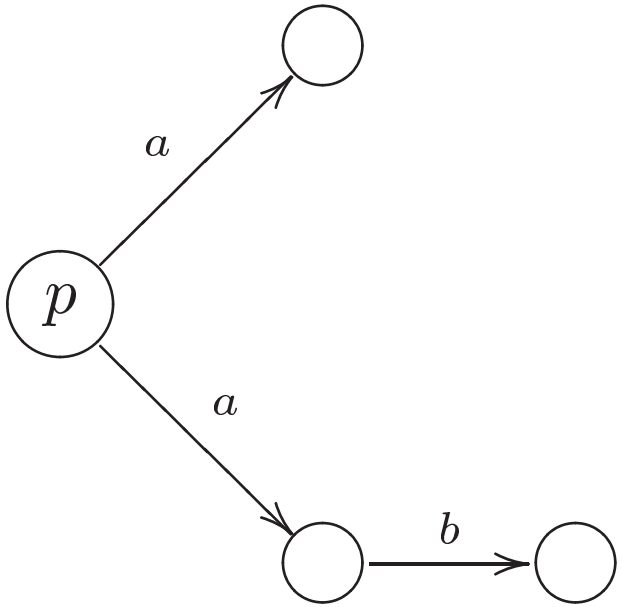
$\sim$ est une relation d'équivalence.

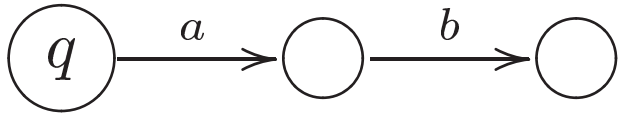


Bissimulation

Attention !

Il n'y a pas bissimulation dans





Bissimulation

Attention !

p simule fortement q et q simule fortement p .

Autrement dit, il existe $\mathcal{S}$ et $\mathcal{S}'$ telles que

– $\mathcal{S}$ et $\mathcal{S}'$ sont des simulations fortes

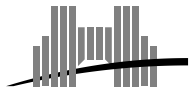
– et $p \; \mathcal{S} \; q$

– et $q \; \mathcal{S}' \; p$,

– mais $\mathcal{S}'$ n'est pas la converse de $\mathcal{S}$.

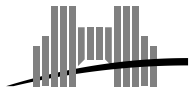
Il n'y a donc pas de bissimulation.

La mobilité



La plasticité ... désigne la capacité générale du neurone et de ses synapses à changer de propriétés en fonction de leur état d'activité. Cette propriété fondamentale va à l'encontre de l'impression naïve selon laquelle le cerveau serait une sorte d'automate rigide constitué exclusivement de «roues et engrenages».

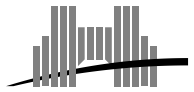
Jean-Pierre Changeux, L'Homme de vérité, Odile Jacob, (2002). p.43



Je le répète parce que c'est un point important, le réseau cérébral ne s'assemble pas comme on construit un ordinateur avec des pièces détachées suivant un plan prédéterminé qui définirait exactement la nature et la disposition de chaque circuit et de chaque commutateur.

Si tel était le cas, une erreur de détail, même minime dans la réalisation de ce programme pourrait avoir des conséquences catastrophiques.

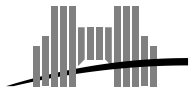
Jean-Pierre Changeux, L'Homme de vérité, Odile Jacob. (2002), p. 284



Il s'agit d'une vision «datée» des ordinateurs.

A l'heure d'Internet, la reconfiguration des architectures des systèmes informatiques est continuelle.

C'est ce que nous allons étudier.



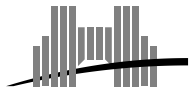
Qu'est-ce que la mobilité ?

On cherche à abstraire les concepts clés de la mobilité,

quand les processus, les programmes, les matériels, les agents migrent établissent et reconfigurent des liens.

Trois sortes de vues de la mobilité :

1. Les processus se déplacent dans un espace physique de sites de calcul.
2. Les processus se déplacent dans un espace virtuel de processus liés.
3. Les liens se déplacent dans un espace virtuel de processus liés.



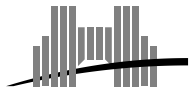
Qu'est-ce que la mobilité ?

Nous allons étudier la troisième vue.

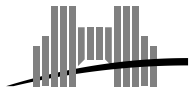
> Les liens se déplacent dans un espace virtuel de processus liés.

Dans le $\pi - calcul$, il n'y a plus que de liens.

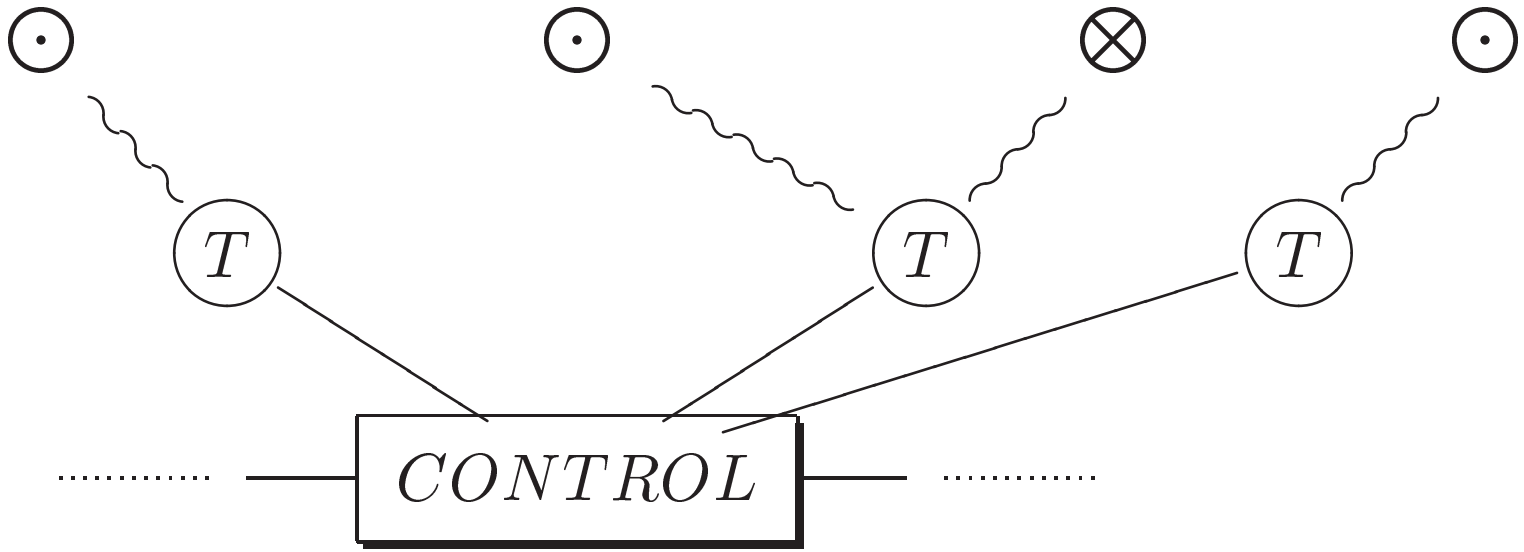
Malgré tout c'est extrêmement puissant.



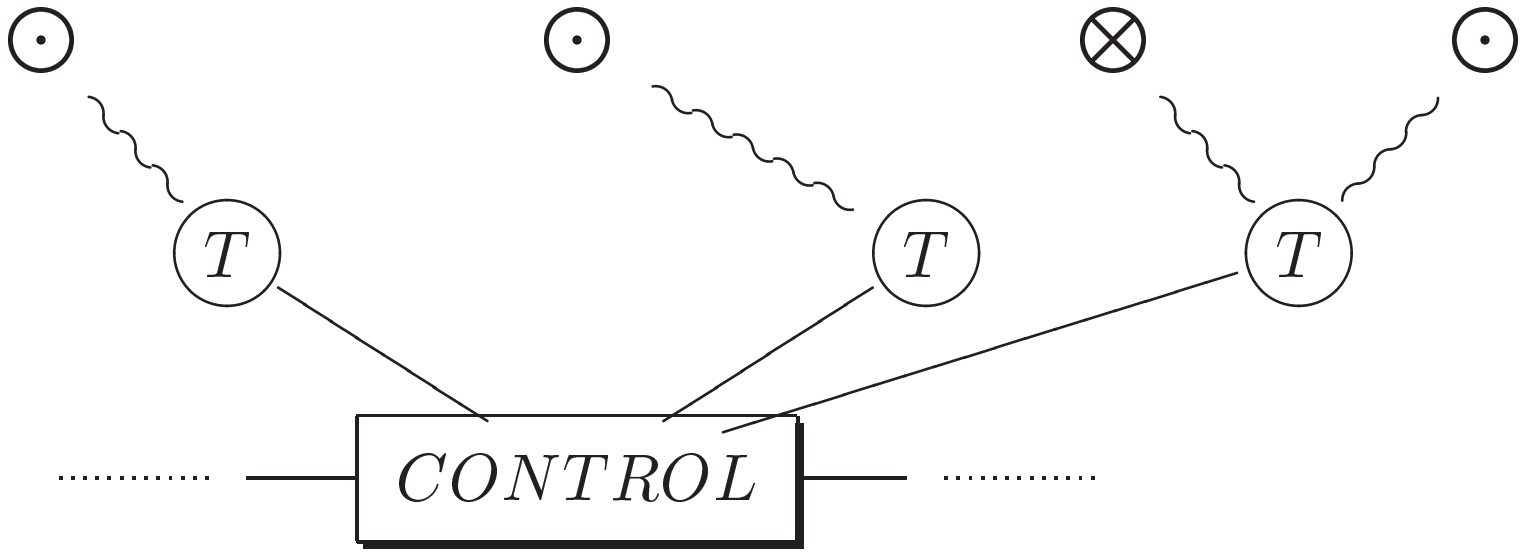
Un exemple : les téléphones mobiles



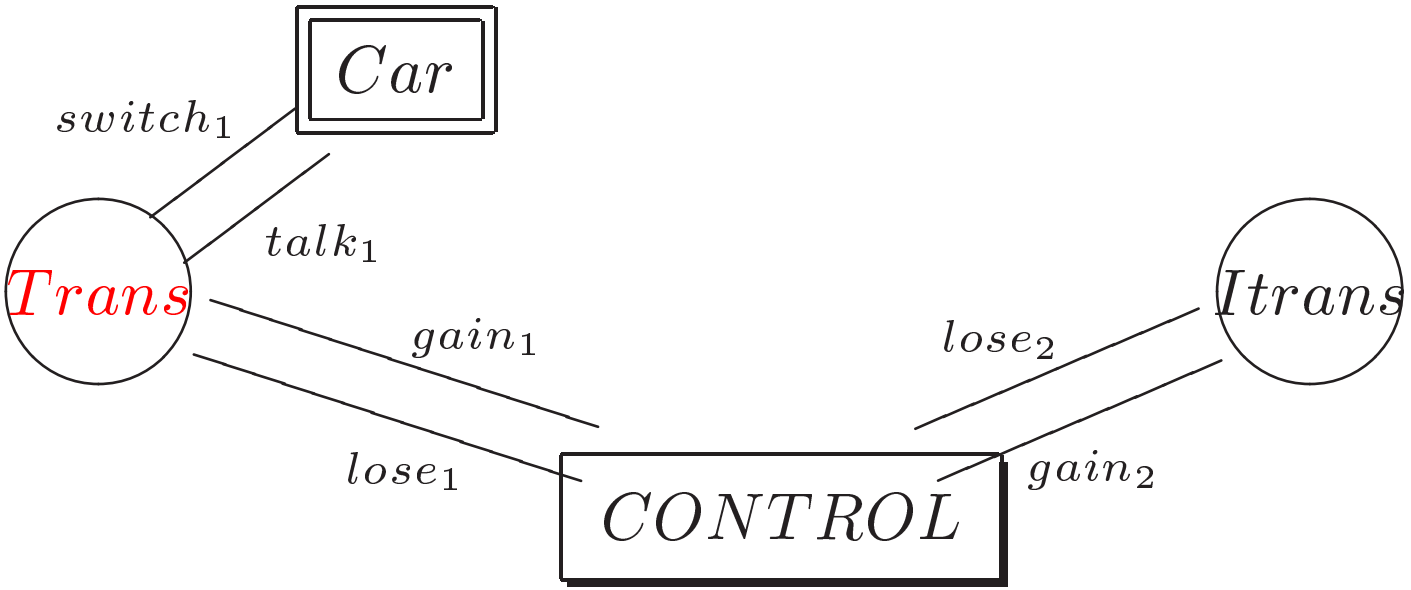
Les téléphones mobiles



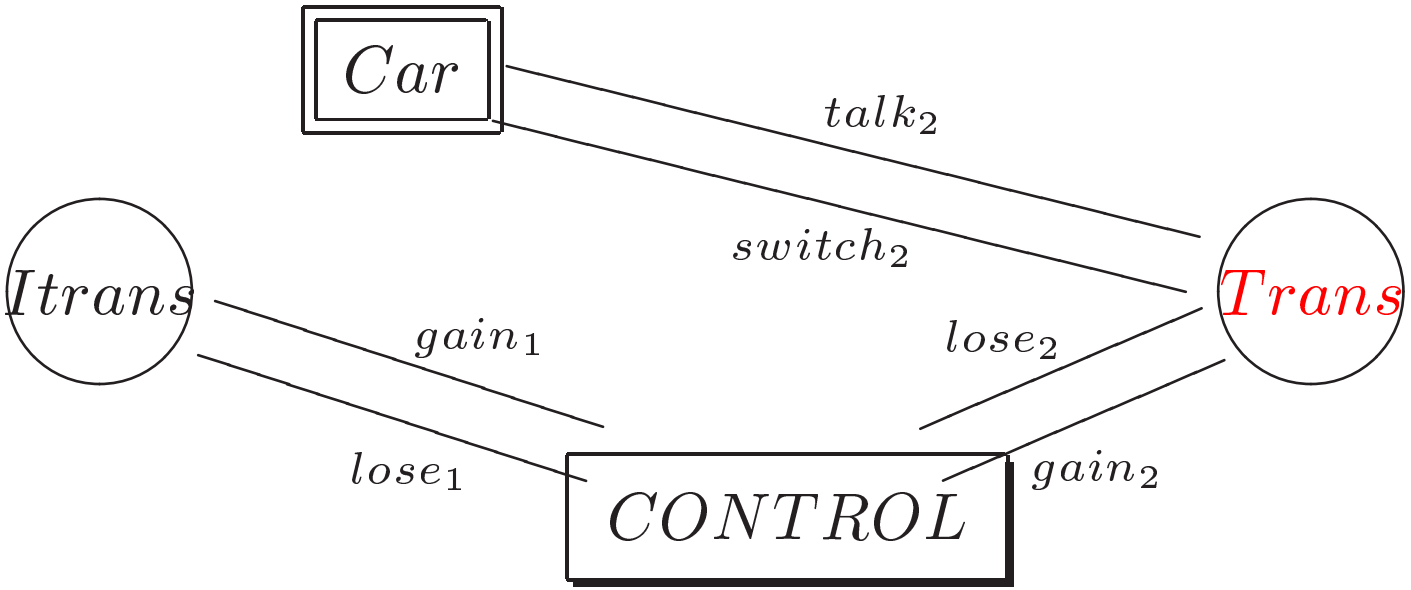
Les téléphones mobiles



Le téléphone mobile à un seul mobile



Le téléphone mobile à un seul mobile



Le téléphone mobile à un seul mobile

Faisons l'hypothèse qu'il n'y a que deux transmetteurs

– un transmetteur actif

– et un transmetteur inactif

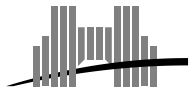
Le transmetteur actif a la forme

$$Trans\langle talk_1, switch_1, gain_1, lose_1\rangle$$

dont les liens courants sont les paramètres de transmission.

Les processus sont notés par des identificateurs commençant par une majuscule.

Les canaux sont notés par des identificateurs commençant par une minuscule.



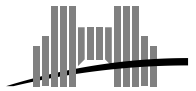
Le téléphone mobile à un seul mobile

La voiture parle au transmetteur sur le canal $talk_1$.

Mais la centrale de contrôle peut demander à tout instant à la voiture par l'intermédiaire du transmetteur de commuter vers un autre transmetteur, dont elle lui transmet les coordonnées :

c'est-à-dire les liens pour communiquer avec lui.

Pour cela,la centrale de contrôle communique avec la transmetteur par le canal $lose_1$.



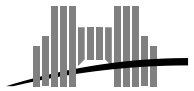
Le téléphone mobile à un seul mobile

On a donc deux équations :

$$Trans(talk, switch, gain, lose) \stackrel{def}{=} talk.Trans\langle talk, switch, gain, lose\rangle + lose(t, s).\overline{switch}\langle t, s\rangle.Itrans\langle gain, lose\rangle$$

$$Itrans(gain, lose) \stackrel{def}{=} gain(t, s).Trans\langle t, s, gain, lose\rangle.$$

Les interactions transmettent des noms de canaux dans leurs messages.

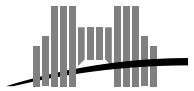


Le téléphone mobile à un seul mobile

Une action positive comme $lose(s, t)$ reçoit des noms.

Une action négative comme $\overline{switch}\langle t, s\rangle$ envoie des noms.

$lose(s, t)$ est un lieur, il introduit des noms locaux comme s et t.



Le téléphone mobile à un seul mobile

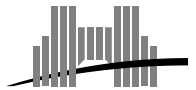
Une action positive comme $lose(s, t)$ reçoit des noms.

Une action négative comme $\overline{switch}\langle t, s\rangle$ envoie des noms.

$lose(s, t)$ est un lieur, il introduit des noms locaux comme s et t.

Les parenthèses $(\ldots)$ servent à lier des variables de noms de canaux reçus.

Les chevrons $\langle\ldots\rangle$ servent à dénoter des tuples de noms de canaux que l'on envoie.



Le téléphone mobile à un seul mobile

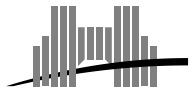
$Control$ qui dit à $Trans$ de lâcher (lose) une voiture doit lui

transmettre les nouveaux canaux pour émettre.

Il doit transmettre les mêmes canaux au transmetteur inactif pour lui permettre d'entrer en contact avec la voiture.

$$Control_1 \stackrel{def}{=} \overline{lose_1}\langle talk_2, swicth_2\rangle.\overline{gain_2}\langle talk_2, switch_2\rangle.Control_2$$

$$Control_2 \stackrel{def}{=} \overline{lose_2}\langle talk_1, swicth_1\rangle.\overline{gain_1}\langle talk_1, switch_1\rangle.Control_1$$

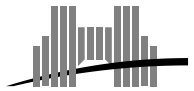


Le téléphone mobile à un seul mobile

La voiture (car) peut

– ou bien parler (talk)
– ou bien échanger (switch) une nouvelle paire de canaux.

$$Car(talk, switch) \stackrel{def}{=\!=} \overline{talk}.Car\langle talk, switch\rangle + switch(t, s).Car\langle t, s\rangle.$$



Le téléphone mobile à un seul mobile

Comment faire de tout cela un processus ?

$$System_1 \stackrel{def}{=} \text{new } talk_1, switch_1, gain_1, lose_1, talk_2, switch_2, gain_2, lose_2$$
$$(Car\langle talk_1, switch_1\rangle \mid Trans_1 \mid Itrans_2 \mid Control_1)$$

avec les fonctions auxiliaires

$$Trans_i \stackrel{def}{=} Trans\langle talk_i, switch_i, gain_i, loase_i\rangle$$

$$Itrans_i \stackrel{def}{=} Itrans\langle gain_i, lose_i\rangle$$

Le téléphone mobile à un seul mobile

À l'aide du π-calcul, on mettra en évidence que ce protocole est correct.

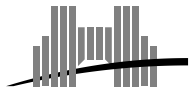
C'est-à-dire que les passage de «relais» et les attributions des canaux de communication se font bien.
Autrement dit que l'on a bien :

$$System_1 \xrightarrow{*} System_2$$

où $System_2$ est juste $System_1$ avec les indices permutés :

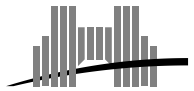
$$System_2 \stackrel{def}{=} \text{new } talk_2, switch_2, gain_2, lose_2, talk_1, switch_1, gain_1, lose_1$$
$$(Car\langle talk_2, switch_2\rangle \mid Trans_2 \mid Itrans_1 \mid Control_2).$$

Le π-calcul



Les actions

$$\begin{array}{lll} \pi & ::= & x(y) \qquad \text{reçoit } y \text{ sur } x \\ & & \overline{x}\langle y \rangle \qquad \text{envoie } y \text{ sur } x \\ & & \tau \qquad \text{action inobservable} \end{array}$$



Le π-calcul

$$P \quad ::= \quad \sum_{i \in I} \pi_i.P_i \;\Big|\; P_1 | P_2 \;\Big|\; \text{new } aP \;\Big|\; !P$$

$\sum_{i \in I} \pi_i.P_i$ est une somme

dans laquelle les P_i sont gardés par les π_i.

0 est la somme vide qui représente l'action vide.

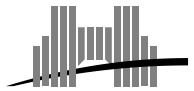
Nous l'omettons après une action.

On écrit $\overline{x}\langle y \rangle$ au lieu de $\overline{x}\langle y \rangle.0$.

La réplication

| s'appelle la composition concurrente.

$!P$ s'appelle la réplication ou le bang.



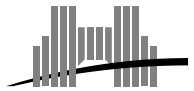
Un exemple

$$P \quad = \quad \text{new } z((\overline{x}\langle y\rangle + z(w).\overline{w}\langle y\rangle) \mid x(u).\overline{u}\langle v\rangle \mid \overline{x}\langle z\rangle).$$

x, y et v sont les seuls noms libres dans P.

Des paires positives et négatives qui utilisent le même canal sont dites être des paires complémentaires.

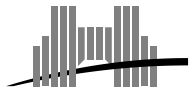
$\overline{x}\langle y\rangle$ et $x(u)$ sont complémentaires.



Redex et déclenchement

Si une paire complémentaire est formée de deux actions situées dans des sommes différentes et si elles sont non gardées alors elles forment un redex.

Le déclenchement de ce redex constitue une réaction $P \to P'$ qui met en œuvre une substitution ici $\{y/u\}$.



Redex et déclenchement

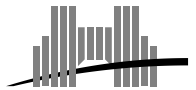
Si une paire complémentaire est formée de deux actions situées dans des sommes différentes et si elles sont non gardées alors elles forment un redex.

Le déclenchement de ce redex constitue une réaction $P \to P'$ qui met en œuvre une substitution ici $\{y/u\}$.

La règle principale est

$$(x(y).Q + M) \mid (\overline{x}\langle z\rangle.R + N) \to \{z/y\}Q \mid R$$

Les autres règles sont contextuelles.



Un exemple

P a deux redex

– la paire $\overline{x}\langle y\rangle$, $x(u)$

– la paire $x(u)$, $\overline{x}\langle z\rangle$

Deux réactions possibles

– $P \to P_1$

– $P \to P_2$

avec

$$P_1 = \text{new } z(0 \mid \overline{y}\langle v\rangle \mid \overline{x}\langle z\rangle)$$

$$P_2 = \text{new } z((\overline{x}\langle y\rangle + z(w).\overline{w}\langle y\rangle) \mid \overline{z}\langle v\rangle \mid 0).$$

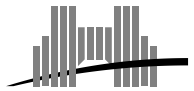
Un exemple

Il n'y a plus de redex dans P_1.

Et il y a la paire $z(w)$ et $\overline{z}\langle v \rangle$ dans P_2.

Nous avons $P_2 \rightarrow P_3$ où

$$P_3 \quad = \quad \text{new } z(\overline{v}\langle y \rangle \mid 0 \mid 0).$$

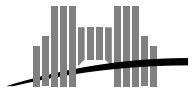


Congruence de processus

$\cong$ est une relation d'équivalence telle que

$$\frac{P \cong Q}{\pi.P + M \cong \pi.Q + M} \qquad \frac{P \cong Q}{\text{new } x.P \cong \text{new } xQ}$$

$$\frac{P \cong Q}{P \mid R \cong Q \mid R} \qquad \frac{P \cong Q}{R \mid P \cong R \mid Q} \qquad \frac{P \cong Q}{!P \cong !Q}$$

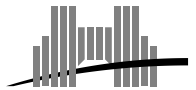


Congruence structurelle

La congruence structurelle $\equiv$ est définie comme la plus petit congruence qui contient :

- l'alpha-conversion,
- l'associativité et la commutativité dans les sommes,
- pour $\mid$, l'associativité, la commutativité, la neutralité de 0,
  $P \mid Q \equiv Q \mid P \quad P \mid 0 \equiv P$,
- new $x(P \mid Q) \equiv P \mid$ new xQ if $x \notin fn(P)$,
  new $0 \equiv 0$, new $x\ y\ P \equiv$ new $y\ x\ P$,
- $!P \equiv P \mid !P$,

$fn(P)$ est l'ensemble de noms libres de P.



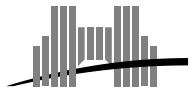
Congruence structurelle

$$\text{new } z((\overline{x}\langle y\rangle + z(w).\overline{w}\langle y\rangle) \mid x(u).\overline{u}\langle v\rangle \mid \overline{x}\langle z\rangle)$$

$$\equiv$$

$$x(u).\overline{u}\langle v\rangle \mid \text{new } z((z(w).\overline{w}\langle y\rangle + \overline{x}\langle y\rangle) \mid \overline{x}\langle z\rangle)$$

$$\text{new } z(\overline{v}\langle y\rangle \mid 0 \mid 0) \quad \equiv \quad \overline{v}\langle y\rangle.$$



Forme standard

Un processus

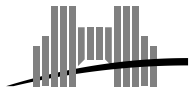
$$\text{new } \vec{a}(M_1 \mid ... \mid M_m \mid !Q_1 \mid ...!Q_n)$$

est en forme standard

– si aucun des M_i n'est une somme nulle

– et si chaque Q_j est lui-même une forme standard.

Chaque processus est équivalent à une forme standard.



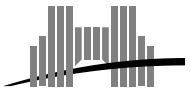
Relation de réaction

$$\frac{}{\tau.P + M \to P}\, TAU$$

$$\frac{}{(x(y).P + M) \mid (\overline{x}\langle z\rangle.Q + N) \to \{z/y\}P \mid Q}\, REACT$$

$$\frac{P \to P'}{P \mid Q \to P' \mid Q}\, PAR \qquad \frac{P \to P'}{\text{new } xP \to \text{new } xP'}\, RES$$

$$\frac{P \to P'}{Q \to Q'}\, STRUCT \qquad \text{si } P \equiv P \text{ et } Q \equiv Q.$$



La récursivité

Le π-calcul ne contient pas de définitions récursives.

Mais on peut facilement les simuler.

Supposons que l'on ait la définition [a]

$$A(x) \quad = \quad Q_A \quad \text{où} \quad Q_A = ...A\langle u\rangle...A\langle v\rangle...$$

qui apparaît dans

$$P \quad = \quad ...A\langle y\rangle...A\langle z\rangle...$$

[a] Je prends une fonction à un seul paramètre, mais ça se généralise.

La récursivité

On transforme la définition en utilisant les réplications.

1. On utilise un nouveau nom, disons a, pour A.
2. On remplace chaque invocation du processus R

   par l'invocation du processus $\widehat{R}$

   obtenu en remplaçant chaque occurrence de $A\langle w \rangle$ par $\overline{a}\langle w \rangle$.
3. On remplace P par

$$\widehat{\widehat{P}} \quad = \quad \text{new } a \, (\widehat{P} \mid !a(x).\widehat{Q_A})$$

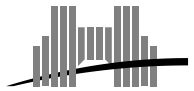
La récursivité

Par exemple,

$$Car(talk, switch) \stackrel{def}{=} \overline{talk}.Car\langle talk, switch\rangle + switch(t, s).Car\langle t, s\rangle.$$

et

$$System_1 \stackrel{def}{=} \text{new } talk_1, switch_1, gain_1, lose_1, talk_2, switch_2, gain_2, lose_2$$
$$(Car\langle talk_1, switch_1\rangle \mid Trans_1 \mid Itrans_2 \mid Control_1)$$



se transforme en[a]

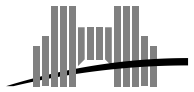
$$System_1 \stackrel{def}{=} \text{new } car$$

$$\text{new } talk_1, switch_1, gain_1, lose_1, talk_2, switch_2, gain_2, lose_2$$

$$(\overline{car}\langle talk_1, switch_1\rangle \mid Trans_1 \mid Itrans_2 \mid Control_1$$

$$\mid !car(lose, swicth).(\overline{talk}.\overline{car}\langle talk, switch\rangle + switch(t, s).\overline{car}\langle t, s\rangle$$

[a] J'utilise des canaux à plusieurs paramètres.



Tampons ou Buffers

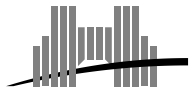
Un buffer qui conserve une valeur peut se représenter par les équations :

$$B(l,r) \stackrel{def}{=} l(x).C\langle x,l,r\rangle$$

$$C(x,l,r) \stackrel{def}{=} \overline{r}\langle x\rangle.B\langle l,r\rangle$$

Tant que la valeur n'est pas réclamée sur le canal r la valeur transmise sur la canal l (notée intérieurement x) est conservée.

Une fois cette valeur transmise, le buffer est prêt pour un autre stockage.



Tampons ou Buffers

Supposons que le processus appelant soit :

$$...B\langle i, o\rangle ...C\langle x, i, o\rangle ...$$

Il se code en π-calcul sous la forme :

$$\text{new } b, c \quad ...(\overline{b}\langle l, r\rangle ...\overline{c}\langle x, l, r\rangle ... \mid !b(i, o).i(x).\overline{c}\langle x, i, o\rangle \mid !c(y, e, s).\overline{e}\langle y\rangle .\overline{b}\langle e, s\rangle).$$

7 variables sont liées dans cette formule : b, c, i, o, y, e, s.

Enchaînement de buffers

Pour allonger la capacité d'un buffer, on peut en «lier» plusieurs

$$B_1 \frown B_2 \ldots \frown B_n$$

L'opération $\frown$ se définit sur des processus, par exemple si F et G sont d'arité 2 (notés $F : 2$ et $G : 2$).

$$(F \frown G)\langle l, r\rangle \quad \stackrel{def}{=\!=} \quad \text{new } m\ (F\langle l, m\rangle \mid G\langle m, r\rangle).$$

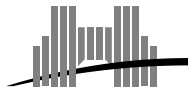
Enchaînement de buffers

Avec $B : 2$ et $C : 3$ on peut écrire $C\langle x\rangle : 2$ et aussi :

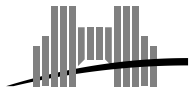
$$J \quad = \quad C\langle x_1\rangle \frown B \frown C\langle x_2\rangle.$$

où

$$J\langle l, r\rangle \quad = \quad \text{new}\; b\, m_1\, m_2\, (\overline{c}\langle x_1, l, m_1\rangle \quad | \quad \overline{b}\langle m_1, m_2\rangle | \overline{c}\langle x_1, m_2, r\rangle$$
$$| \quad !b(i, o).i(x).\overline{c}\langle x, i, o\rangle \quad | \quad !c(y, e, s).\overline{e}\langle y\rangle.\overline{b}\langle e, s\rangle)$$



Systèmes simples



Systèmes simples

Un système en forme standard

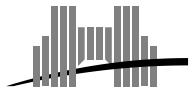
$$P \quad \equiv \quad \text{new } \vec{z}(M_1 \mid ... \mid M_m \mid !N_1 \mid ... \mid !N_n)$$

est simple si

S1 P est structurellement équivalent à un système dans lequel toutes les réplications sont des sommes.

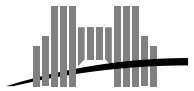
S2 Aucun M_i, ni aucun N_j ne contient de réplication.

S3 Aucun M_i, ni aucun N_j ne contient de composition concurrente.



Systèmes simples

La simplicité est stable par réaction.



Systèmes simples

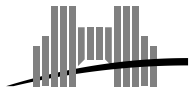
Les buffers sont simples.

On voit que

new $b\ m_1\ m_2$ $\quad (\overline{c}\langle x_1, l, m_1\rangle \mid \overline{b}\langle l, r\rangle \mid \overline{c}\langle x_1, m_2, r\rangle$

$\quad \mid\ !b(i,o).i(x).\overline{c}\langle x, i, o\rangle \mid\ !c(y,e,s).\overline{e}\langle y\rangle.\overline{b}\langle e, s\rangle)$

est simple.



Propriétés de systèmes

On veut prouver des propriétés sur les systèmes.

Ainsi dans le système des téléphones mobiles on veut pouvoir s'assurer que la centrale de contrôle n'a pas attribué le même canal de communication à plus d'un mobile[a].

[a] Ça m'est arrivé sur un téléphone fixe, c'est désagréable !

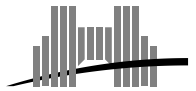
Propriétés de systèmes

Dans un système complexe ça peut être vraiment compliqué.

On veut pourvoir mettre cela en évidence à travers des propriétés syntaxiques.

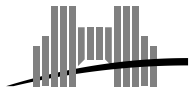
Et on veut pouvoir prouver des théorèmes à propos des systèmes.

L'exigence de simplicité est un cadre qui peut permettre de prouver sans trop de contorsion des théorèmes suffisamment riches.



Prise

P a une prise sur x si P a une occurrence libre de x dans une action de la forme $\overline{x}\langle \ldots \rangle$ ou $\overline{v}\langle \ldots x \ldots \rangle$.



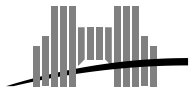
Prise unique

Définition : prise unique

Si P est un système simple en forme standard,

on dit que P a une prise unique sur x

– si au plus un seul M_i a une prise sur x
– et aucun N_j n'a une prise sur x.



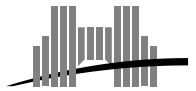
Abandon de x

Définition : abandon de x

Un terme P abandonne x si dans chaque terme de la forme $a\langle\vec{u}x\vec{v}\rangle.Q$

– ni $\vec{u}$, ni $\vec{v}$ ne contiennent x,

– Q n'a pas prise sur x,

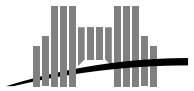


Préservation de la prise unique

Soit P un système simple qui satisfait les conditions suivantes pour un certain x

1. P a une prise unique sur x,
2. P abandonne x,
3. pour chaque sous-terme de la forme $z(\vec{y}).Q$,
   Q abandonne chacun des $y \in \vec{y}$

Si $P \to P'$, alors P' est simple et satisfait 1., 2. et 3.



Démonstration du théorème

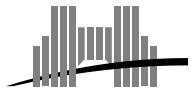
On la fait dans la cas monadique.

On suppose

$$P \quad \equiv \quad \text{new } \vec{z}(M_1 \mid ... \mid M_m \mid !N_1 \mid ... \mid !N_n)$$

Sans nuire à la généralité on peut supposer que l'interaction se fait entre M_1 et M_2.

Par exemple, on peut «décomposer» $!N_j$ en $N_j \mid !N_j$ et prétendre que N_j est M_1 ou M_2.



Démonstration du théorème

Donc

$$M_1 \equiv \ldots + \overline{u}\langle v\rangle \text{new } \vec{z_1}\, M_1'$$
$$M_2 \equiv \ldots + u(w).\text{new } \vec{z_2} M_2'$$

et

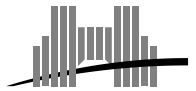
$$P' \equiv \text{new } \vec{z_1}\ \vec{z_2}(M_1' \mid \{v/w\}M_2' \mid \ldots \mid M_m \mid !N_1 \mid \ldots \mid !N_n)$$

Démonstration du théorème

P' a une prise unique

On va montrer qu'il y a moins de prises sur x dans $M_1' \mid \{v/w\}M_2'$
qu'il y en a dans $M_1 \mid M_2$.

- M_1' a moins de prises sur x que M_1,
- $\{v/w\}M_2'$ ne crée pas plus de prises sur x.
  - Si $v \neq x$ alors $\{v/w\}M_2'$ et M_2 ont les mêmes prises sur x,
  - Si $v = x$ alors l'unique prise de x dans P est $\overline{u}\langle v\rangle$.new $\vec{z_1}\,M_1'$, mais d'après la propriété d'abandon, M_1' n'a pas de prise sur x. Donc $\{v/w\}M_2'$ (alias $\{x/w\}M_2'$) peut avoir prise sur x en préservant l'unicité de la prise.



Démonstration du théorème

P' abandonne x

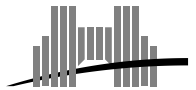
L'abandon est une propriété sur des sous-termes qui ne peut changer que si les dits sous-termes changent.

Donc, clairement, seul $\{v/w\}M_2'$ et encore dans le cas où $v = x$ pourrait ne pas abandonner x.

Supposons donc que $\{x/w\}M_2'$ n'abandonne pas x.

Cela veut dire que M_2' n'abandonne pas w,

autrement dit que $u(w)$.new $\vec{z_2}M_2'$ contredit la condition (3).



Démonstration du théorème

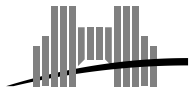
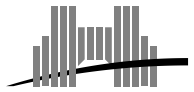
Condition (3)

La condition (3) ne peut être violée dans P' que dans $\{v/w\}M_2'$.

Supposons que ce soit dans $z'(y).\{v/w\}Q$ (z' peut être w ou non).

Alors elle est violée dans $z(y).Q$ (z est v si z' est w et est z' sinon).

Ce qui contredit le fait que P satisfait (3).



Et voilà démontré un théorème dans le π-calcul !