

CCS

a Calculus of Communicating Systems

Pascal Poizat

<http://www.lami.univ-evry.fr/~poizat/enseignement-fr.php>

Université d'Evry Val d'Essonne

Maîtrise Informatique, Génie Logiciel 2

2003 – 2004

D'après :

- une version en anglais de ces transparents :
CSE 635 / Asynchronous Systems - Lecture Notes,
par R. Cleveland et S. A. Smolka.
 - des mêmes auteurs, l'article *Process Algebra*,
Encyclopedia of Electrical Engineering, John Wiley
and Sons, 1999.
- les deux accessibles depuis ma page web !

- CCS : une algèbre de processus (AP) simple
- Syntaxe
- Sémantique
- Outils théoriques :
 - équivalences
 - ordres
 - (axiomatisations)
- ⇒ permettre la spécification et vérification avec CWB-NC

- Nombreux langages :
 - Basiques : CCS, CSP, Basic LOTOS
 - Extensions données : PSF, FSP, LOTOS
 - Extensions/dialectes avec temps, priorités, probas...
- Leur point commun : approche pour spécifier et vérifier la dynamique des systèmes (réactifs) en concurrence entrelacée (lg. Asynchrones)
- De + en + utilisées (syst. Réactifs, embarqués, langages de composants OO / ADL)

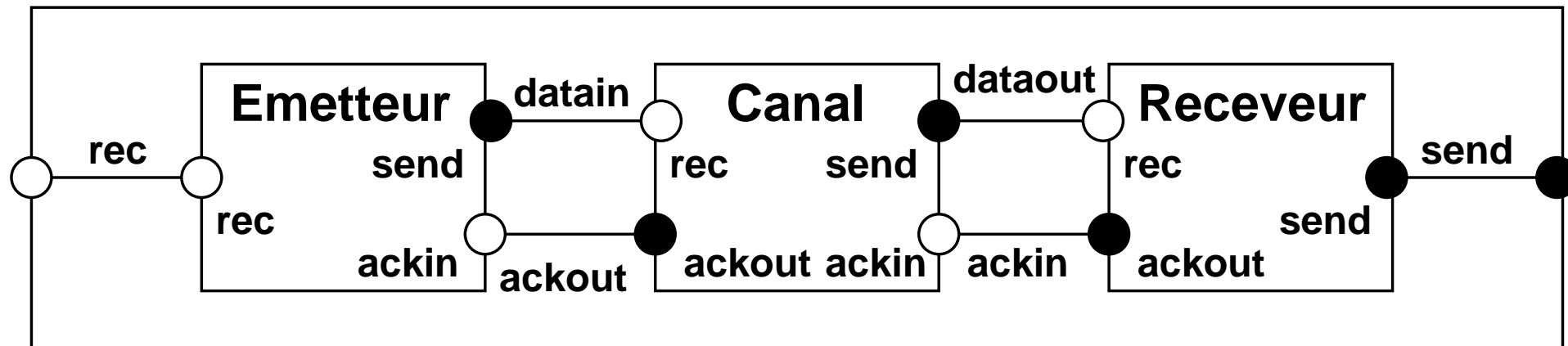
- Systèmes ouverts
- Théories construites autour des notions d'interactions entre systèmes et environnements
- Concurrency (entrelacement)
- Compositionnalité
 - Si $P = Q$ alors pour tout contexte $C[\dots]$, $C[P] = C[Q]$
 - Autres interprétation en termes de validation de propriétés $P = Q \Rightarrow (P \models \Phi \Rightarrow Q \models \Phi)$

- Un langage de spécification qui comprend la possibilité d'exprimer des interactions puis évolution, et des opérateurs pour assembler sous-systèmes en systèmes
- Une sémantique (opérationnelle) formelle basée sur des interactions atomiques syst./envt.
- Une notion de raffinement comportemental

- Décrire le comportement de haut niveau désiré

$$ABP_{SPEC} = rec.\overline{send}.ABP_{SPEC}$$

- Décrire un système candidat



- Montrer la correction du candidat p/r spéc.

- Décrire le comportement de haut niveau désiré

$$ABP_{SPEC} = \overline{rec.send}.ABP_{SPEC}$$

- Décrire un système candidat

$$Emetteur = \overline{rec.send}.ackin.Emetteur$$

$$Canal = \overline{rec.send}.Canal$$

$$+ \overline{ackin.ackout}.Canal$$

$$Receveur = \overline{rec.send}.\overline{ackout}.Receveur$$

$$ABP = (Emetteur[datain/send, ackout/ackin] \\ | Canal[datain/rec, dataout/send] \\ | Receveur[dataout/rec, ackin/ackout] \\) \setminus \{datain, dataout, ackin, ackout\}$$

- Montrer la correction du candidat p/r spéc.

- Représentant simple des AP
(CCS,CSP,SCCS,TCSP,ACP,LOTOS,...)
- développé par Milner (Turing Award) fin 70, déb. 80
- Mécanisme d'interaction de base :
rdz-vs/synchronisation binaire a/\bar{a}
(mais nombreuses extensions existent ...)
- Spécification : processus construits à partir d'actions atomiques et à partir d'opérateurs

- Entrées/sorties (+ généralement synchronisations) sur ports de communication
- Λ : ensemble (ev. ∞) d'étiquettes (alphabet)
- Actions :
 - Λ : ensemble d'étiquettes et ensemble des actions d'entrée
 - $\bar{\Lambda} = \{\bar{\lambda} | \lambda \in \Lambda\}$: ensemble des actions de sortie
 - $\Lambda \cup \bar{\Lambda}$: ensemble des actions externes
 - τ : action interne
 - $Act = \Lambda \cup \bar{\Lambda} \cup \{\tau\}$: ensemble des actions
- Remarque : les étiquettes induisent des modèles et logiques étendus (ST/LTS, CTL/ACTL,...)

Les systèmes CCS (**processus**) communiquent avec leur **environnement** (et entre eux) en se **synchronisant** sur des ports

- si un partenaire peut recevoir et l'autre émettre sur **le même port** alors il y a synchronisation et les deux évoluent
- réceptions et émissions sont **blocantes**
- seule action autonome : τ
 - ➡ actions externes d'un processus : son interface

$a \in \text{Act}, L \subseteq \Lambda, f : \Lambda \rightarrow \Lambda$

\mathcal{C} ensemble de noms de processus, $C \in \mathcal{C}$

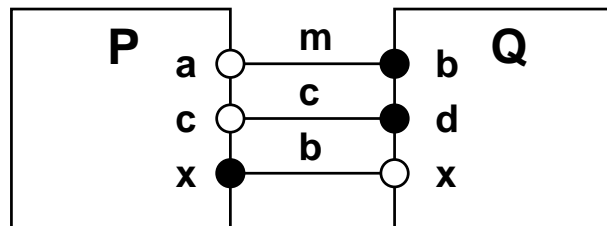
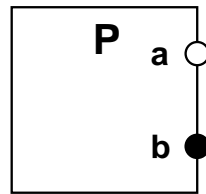
$E ::=$	0	stop
	$ $	$a.E$ préfixe
	$ $	$E_1 + E_2$ choix
	$ $	$E_1 E_2$ parallèle
	$ $	$E \setminus L$ restriction
	$ $	$E[f]$ renommage
	$ $	C invocation

- expressions close (tout processus est déclaré)

- déclaration : $C = E$

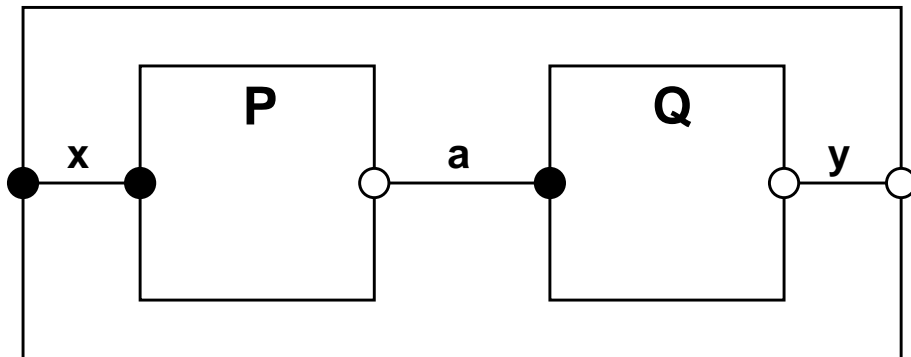
- \mathcal{P} : ens. des processus CCS = ens. des expr. CCS

(interaction)



Processus P
avec alphabet $\{a, b\}$

Composition parallèle
 $(P[m/a] || Q[m/b, c/d])$



Processus composite
 $S = (P|Q) \setminus \{a\}$

Rappel : la communication est binaire synchrone donc ne pas masquer peut poser pb.

- définir le comportement d'expressions CCS en donnant une **sémantique opérationnelle** à CCS
- elle définit les pas d'exécution des systèmes CCS
 - ⇒ modèles étendus, propriétés étendues
- elle sera aussi à la base de l'étude des équivalences comportementales

“Presse-bouton”

- systèmes : boîte avec boutons = actions visibles
- 2 types de boutons (E/S)
- dans différents états, différents boutons possibles

Opérateurs ...

- (interaction)

Relation ternaire $\longrightarrow \subseteq \mathcal{P} \times \text{Act} \times \mathcal{P}$

- $\langle P, a, Q \rangle \in \longrightarrow$: P propose a puis se comporte comme Q après a faite
- notation : $P \xrightarrow{a} Q$
- induit un LTS (Labelled Transition System)

- donnée **inductivement** (SOS) : collection de règles
- forme $\text{nom} \frac{\text{Hypotheses}}{\text{Conclusion}} \text{condition}$
où hypothèses et conclusions sont de la forme
 $P \xrightarrow{a} Q$
- objectif : $P \xrightarrow{a} Q$ vrai si dérivable à partir des règles

$$\text{Act} \frac{}{a.P \xrightarrow{a} P}$$

$$\text{Sum1} \frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'}$$

$$\text{Sum1} \frac{Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} Q'}$$

$$\text{Com1} \frac{P \xrightarrow{a} P'}{P|Q \xrightarrow{a} P'|Q}$$

$$\text{Com2} \frac{Q \xrightarrow{a} Q'}{P|Q \xrightarrow{a} P|Q'}$$

$$\text{Com3} \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$\text{Res} \frac{P \xrightarrow{a} P'}{P \setminus L \xrightarrow{a} P' \setminus L} \quad a \notin L, \bar{a} \notin L$$

$$\text{Rel} \frac{P \xrightarrow{a} P'}{P[f] \xrightarrow{\hat{f}(a)} P'[f]}$$

$$\text{Con} \frac{P \xrightarrow{a} P'}{C \xrightarrow{a} P'} \quad C \in \mathcal{C}, C = P$$

- $a \in \text{Act}, \bar{\bar{a}} = a, \bar{\tau} = \tau$

- $f : \Lambda \rightarrow \Lambda, \hat{f} : \text{Act} \rightarrow \text{Act}$

- $\hat{f}(a) = \overline{f(b)}$ si $a = \bar{b}$ et $b \in \Lambda$
 τ si $a = \tau$

Un système de transitions étiquettées (LTS) est un système de transitions (S, S_0, T) où $T \subseteq S \times \text{Act} \times S$.
Relation règles SOS / \longrightarrow ?

- les règles définissent un système d'inférence où les faits inférés ont la forme $P \xrightarrow{a} Q$
 - une transition $P \xrightarrow{a} Q$ peut être inférée s'il est possible de construire une preuve en utilisant les règles
- ⇒ donc \longrightarrow contient **exactement** les $\mathcal{P} \times \text{Act} \times \mathcal{P}$ triplets qui peuvent être inférés à partir des règles

(interaction)

- CCS \Rightarrow LTS
 - états : termes clos
 - transitions : \longrightarrow (SOS)
- LTS \Rightarrow CCS
 - états \rightarrow noms de processus
 - état S donne $S = +_i a_i . S_i$
 - l'encodage d'un LTS ne demande que les opérateurs **dynamiques** (0 , $a.P$ et $+$) et les déclarations
 - les opérateurs **statiques** ($|$, $\backslash L$, $[f]$) fournissent une information architecturale

- ST, Automates, LTS, ... : nombreux, au min. graphique.
Sémantique : MEC, partie CADP
- dédié AP simples (dont CCS) : CWB-NC
`cwb-nc ccs -gui &`
 - animation
 - equivalences
 - model checking
 - recherche de blocage (dealock)
- pour aller plus loin : FSP (AP plus expressive) et son outil, LTSA
`java -jar $LTSAHOME/lib/ltsa.jar`

Soient $P = a.(b.0 + c.0)$ et $Q = a.b.0 + a.c.0$

Mêmes traces : $\{\varepsilon, a, ab, ac\}$, $P =_L Q$

mais ce est-ce ce qui est voulu dans une théorie basée sur les interactions ?

Soit un processus de test $T = \bar{a}.\bar{b}.\bar{\omega}.0$. Ici ω indique le succès. Pour savoir si le test réussit, on le compose avec le système.

$$E1 = (P|T) \setminus \{a, b, c\}, \quad E2 = (Q|T) \setminus \{a, b, c\}$$

$E1 \xrightarrow{*} \bar{\omega}.0$: toujours !, $E2 \xrightarrow{*} \bar{\omega}.0$: pas toujours !

Que faire ?

- ⇒ étude des bissimulations et équivalences associées
- remarque : il existe d'autres équivalences, voir treillis dans Handbook of PA

Milner 89 : Une **bissimulation** est une forme d'invariant entre paires de systèmes dynamiques, et la technique est de prouver deux systèmes équivalents en établissant un tel invariant, de façon très similaire à la façon dont il est possible de prouver la correction d'un programme séquentiel unique en trouvant une propriété d'invariant.

Une relation binaire $S \subseteq \mathcal{P} \times \mathcal{P}$ est une **bissimulation forte** si :

$(P, Q) \in S \Rightarrow \forall a \in \text{Act}$

■ $P \xrightarrow{a} P' \Rightarrow \exists Q', Q \xrightarrow{a} Q' \wedge (P', Q') \in S$

■ $Q \xrightarrow{a} Q' \Rightarrow \exists P', P \xrightarrow{a} P' \wedge (P', Q') \in S$

P et Q sont **fortement équivalents** ou **fortement bissimilaires** ce qui est noté $P \sim Q$ si et seulement s'il existe une bissimulation forte S telle que $(P, Q) \in S$.

$\sim = \bigcup \{S \mid S \text{ est une bissimulation forte}\}$

\sim est une congruence

On trouve aussi parfois la définition suivante dans la littérature : $P \sim Q$ ssi

- $P \xrightarrow{a} P' \Rightarrow \exists Q', Q \xrightarrow{a} Q' \wedge P' \sim Q'$

- $Q \xrightarrow{a} Q' \Rightarrow \exists P', P \xrightarrow{a} P' \wedge P' \sim Q'$

Attention aussi à la similarité : $P \text{ sim } Q \wedge Q \text{ sim } P \not\Rightarrow P \sim Q$

! [On reviendra dessus en TD]

- Prouver $P \sim Q$: trouver (construire) une bissimulation forte contenant (P, Q)
- Prouver $P \not\sim Q$: montrer qu'il n'y en a pas \rightarrow preuve par contradiction
 - supposer qu'il existe une b.f. entre P et Q
 - montrer que cela conduit à une contradiction

$$a.b.0 \not\sim a.b.0 + a.0$$

$$a.(b.0 + c.0) \not\sim a.b.0 + a.c.0$$

$$a.c.0 + b.0 \sim (a.c.0 + b.0) + a.c.0$$

(preuves interactives + arbres utilisés)

Vient du fait que \sim est trop sensible à τ

Ex : $a.\tau.b.0 \not\sim a.b.0$

Idée : introduire une relation de transition dérivée, \Longrightarrow , qui fait abstraction des calculs “internes” (actions internes ou synchronisations internes)

\Longrightarrow , relation de transition faible :

- $P \xRightarrow{\varepsilon} Q$ ssi $P \xrightarrow{\tau} \dots \xrightarrow{\tau} Q$

- $P \xrightarrow{a} Q$ ssi $\exists P', Q' . P \xRightarrow{\varepsilon} P' \xrightarrow{a} Q' \xRightarrow{\varepsilon} Q$

i.e. $P \xrightarrow{a} Q$ ssi $P \xrightarrow{\tau} \dots \xrightarrow{\tau} \xrightarrow{a} \xrightarrow{\tau} \dots \xrightarrow{\tau} Q$

- \hat{a} est ε si $a = \tau$ et a sinon.

- si $s \in \text{Act}^*$ alors $\hat{s} \in (\text{Act} - \{\tau\})^*$ est s où on enlève τ

- $q \Longrightarrow q'$ si $\exists s' \text{ tq } q \xrightarrow{s'} q' \wedge s = \hat{s}'$

Une relation $S \subseteq \mathcal{P} \times \mathcal{P}$ est une bissimulation faible si, quand $\langle P, Q \rangle \in S$ alors :

$$\blacksquare P \xrightarrow{a} P' \Rightarrow \exists Q', Q \xRightarrow{\hat{a}} Q' \text{ tq } \langle P', Q' \rangle \in S$$

$$\blacksquare Q \xrightarrow{a} Q' \Rightarrow \exists P', P \xRightarrow{\hat{a}} P' \text{ tq } \langle P', Q' \rangle \in S$$

P est observationnellement équivalent à Q , noté $P \approx Q$, ssi il existe une bissimulation (faible) S avec $\langle P, Q \rangle \in S$.

Note : les formes de \sim et \approx sont proches donc preuves et commentaires sur simulation idem.

$$a.\tau.b.0 \approx a.b.0$$

$$a.0 + \tau.b.0 \not\approx a.0 + b.0$$

$$ABP_{SPEC} \approx ABP$$

- les +
 - élimine les problèmes de $=_L$
 - insensible (relativement) à τ , élimine les problèmes de \sim
 - techniques de preuves élégantes héritées de \sim

- les +
 - élimine les problèmes de $=_L$
 - insensible (relativement) à τ , élimine les problèmes de \sim
 - techniques de preuves élégantes héritées de \sim
- les -
 - \approx n'est pas une congruence

■ les +

- élimine les problèmes de $=_L$
- insensible (relativement) à τ , élimine les problèmes de \sim
- techniques de preuves élégantes héritées de \sim

■ les -

- \approx n'est pas une congruence

Congruence : une relation d'équivalence est une **congruence** si on peut substituer égal par égal

$$P = Q \Rightarrow \forall C[\dots], C[P] = C[Q]$$

Et alors ?

- les +
 - élimine les problèmes de $=_L$
 - insensible (relativement) à τ , élimine les problèmes de \sim
 - techniques de preuves élégantes héritées de \sim
- les -
 - \approx n'est pas une congruence

Et alors ?

On veut compositionnalité et raisonnement compositionnel / par parties.

En CCS, \approx congruence impliquerait que si $p \approx q$ alors :

En CCS, \approx congruence impliquerait que si $p \approx q$ alors :

- $\forall a \in \text{Act}, a.p \approx a.q$
- $\forall r \in \mathcal{P}, p + r \approx q + r$
- $\forall r \in \mathcal{P}, p|r \approx q|r$
- $\forall L \subseteq \Lambda, p \setminus L \approx q \setminus L$
- $\forall f : \Lambda \rightarrow \Lambda, p[f] \approx q[f]$

En CCS, \approx congruence impliquerait que si $p \approx q$ alors :

- $\forall a \in \text{Act}, a.p \approx a.q$
- $\forall r \in \mathcal{P}, p + r \approx q + r$
- $\forall r \in \mathcal{P}, p|r \approx q|r$
- $\forall L \subseteq \Lambda, p \setminus L \approx q \setminus L$
- $\forall f : \Lambda \rightarrow \Lambda, p[f] \approx q[f]$

tous ok, sauf +

$\tau.b.0 \approx b.0$ mais $\tau.b.0 + a.0 \not\approx b.0 + a.0$

Le problème est lié aux τ donc définition différente de \approx ?

$P \approx^C Q$ ssi $\forall a \in \text{Act}$:

$$\blacksquare P \xrightarrow{a} P' \Rightarrow \exists Q', Q \xRightarrow{a} Q' \wedge P' \approx Q'$$

$$\blacksquare Q \xrightarrow{a} Q' \Rightarrow \exists P', P \xRightarrow{a} P' \wedge P' \approx Q'$$

$$\tau.b.0 \not\approx^C b.0$$

$$a.\tau.b.0 \approx^C a.b.0$$

- si $P \approx Q$, $P \not\xrightarrow{\tau}$ et $Q \not\xrightarrow{\tau}$, alors $P \approx^C Q$
- si $P \approx Q$ alors soit $P \approx^C Q$, $\tau.P \approx^C Q$ ou $P \approx^C \tau.Q$

Ce n'est pas une bidouille :

\approx^C est la plus large congruence contenue dans \approx , ie.

- $P \approx^C Q \Rightarrow P \approx Q$ ($\approx^C \subseteq \approx$)
 - pour toute autre congruence $\approx^D \subseteq \approx$, $\approx^D \subseteq \approx^C$
- ⇒ \approx^C est la congruence la plus “permissive” cohérente par rapport à \approx

- pb lié à $+$. C'est $+$ le vrai pb selon certains
- d'un autre côté, dans la plupart des cas où on utilise le raisonnement compositionnel, on est dans le contexte des opérateurs statiques de CCS, ie on ne substitue pas sous $+$
 - ⇒ \approx est utilisé dans la plupart des cas

Retour sur outils ... CWB-NC.

Si $P \neq Q$ alors contre exemple, ex : $[[a]] \ll b \gg tt$.

Une telle formule existe toujours mais que sont-elles (sorte de LTL ou CTL avec étiquettes) et pourquoi existent-elles ?

Hennessy-Milner Logic

Permet d'écrire des formules modales simples
(pas points fixes)

Caractérise \sim (prouvé par H&M) :

■ $P \sim Q \Leftrightarrow (\forall \phi_{HML}, P \models \phi_{HML} \Leftrightarrow Q \models \phi_{HML})$

■ ou, si $\mathcal{L}(P) = \{\phi_{HML} \mid P \models \phi_{HML}\}$
alors $P \sim Q \Leftrightarrow \mathcal{L}(P) = \mathcal{L}(Q)$

⇒ si $P \not\sim Q$ alors il existe une formule qui les différencie

$a \in \text{Act}$

$\phi ::= tt$	vrai
$\neg\phi$	negation
$\phi_1 \vee \phi_2$	disjonction/union
$\langle a \rangle \phi$	next, existence suivant

$\Phi = \{\phi_{HML}\}$

Logique modale temporelle à actions

(de même qu'il existe aussi ACTL, TLA)

$$\models \subseteq \mathcal{P} \times \Phi$$

notation : $P \models \phi$ au lieu de $\langle P, \phi \rangle \in \models$
 P rend ϕ vraie, ou P valide ϕ

$$P \models tt \quad \forall P \in \mathcal{P}$$

$$P \models \neg\phi \quad \text{si } P \not\models \phi$$

$$P \models \phi_1 \vee \phi_2 \quad \text{si } P \models \phi_1 \text{ ou } P \models \phi_2$$

$$P \models \langle a \rangle \phi \quad \text{si } \exists P', P \xrightarrow{a} P' \text{ et } P' \models \phi$$

dérivés :

$$ff \equiv \neg tt, \phi_1 \wedge \phi_2 \equiv \neg(\neg\phi_1 \vee \neg\phi_2),$$

$$[a]\phi \equiv \neg \langle a \rangle \neg\phi.$$

(interactif)

HML :

- pas très expressive (les formules ne peuvent parler que d'une partie finie des processus)

- MAIS (rappel)

Déf. $P =_{HML} Q \Leftrightarrow (\forall \phi \in \Phi, P \models \phi \Leftrightarrow Q \models \phi)$

Résultat : $P \sim Q \Leftrightarrow P =_{HML} Q$

modulo restriction mineure et technique sur P et Q
(image finite)

- Donner une formule HML distinguant P et Q permet de prouver $P \not\sim Q$

- exemples :

- $a.b.0$ et $a.c.0 \rightarrow ?$

- $a.(b.0 + c.0)$ et $a.b.0 + a.c.0 \rightarrow ?$

Weak HML

$\sim / \text{HML} \rightarrow \approx / \text{wHML}$

fait : \approx est la relation la plus grande telle que, $\forall a \in \text{Act}$:

$$\blacksquare P \xrightarrow{\hat{a}} P' \Rightarrow \exists Q', Q \xrightarrow{\hat{a}} Q' \wedge P' \approx Q'$$

$$\blacksquare Q \xrightarrow{\hat{a}} Q' \Rightarrow \exists P', P \xrightarrow{\hat{a}} P' \wedge P' \approx Q'$$

→ modalité faible : $\langle\langle a \rangle\rangle$

$P \models \langle\langle a \rangle\rangle \phi$ si $\exists P', P \xrightarrow{\hat{a}} P'$ et $P' \models \phi$

$[[a]]\phi \equiv \neg \langle\langle a \rangle\rangle \neg\phi$

Déf. $P =_{\text{wHML}} Q$ de façon analogue à $P =_{\text{HML}} Q$

Résultat : $P \approx Q \Leftrightarrow P =_{\text{wHML}} Q$

Les algos de \sim et \approx ne seront pas vus ici.

MAIS

Il marchent à priori pour des **systemes finis** (nb. états), en effet, il faut calculer $S(P, Q)$.

Les algos de \sim et \approx ne seront pas vus ici.

MAIS

Il marchent à priori pour des **systemes finis** (nb. états), en effet, il faut calculer $S(P, Q)$.

et pour les systemes infinis ?

Les algos de \sim et \approx ne seront pas vus ici.

MAIS

Il marchent à priori pour des **systemes finis** (nb. états), en effet, il faut calculer $S(P, Q)$.

et pour les systemes infinis ?

- construire les bissimulations à la main

Les algos de \sim et \approx ne seront pas vus ici.

MAIS

Il marchent à priori pour des **systemes finis** (nb. états), en effet, il faut calculer $S(P, Q)$.

et pour les systemes infinis ?

- construire les bissimulations à la main
- utiliser un système de preuve équationnel pour prouver les équivalences

Objectif : établir $P = Q$

Ces s.p. combinent **axiomes** et **règles d'inférence** pour permettre des preuves.

Objectif : établir $P = Q$

Ces s.p. combinent **axiomes** et **règles d'inférence** pour permettre des preuves.

Dans le même esprit que (cf maternelle !) :

$$\begin{aligned} 5 + (3 \times 8) + 11 &= 5 + 24 + 11 \\ &= 29 + 11 \\ &= 40 \end{aligned}$$

Objectif : établir $P = Q$

Ces s.p. combinent **axiomes** et **règles d'inférence** pour permettre des preuves.

Dans le même esprit que (cf maternelle !) :

$$\begin{aligned} 5 + (3 \times 8) + 11 &= 5 + 24 + 11 \\ &= 29 + 11 \\ &= 40 \end{aligned}$$

Comment procéder ?

⇒ axiomatisation, liée au **langage** (CCS complet) et à l'**égalité voulue**

⇒ = [\sim , Full CCS]

Pour commencer, **Basic CCS** : pas de $|$, de $\setminus L$, de $[f]$, de constantes de processus.

⇒ juste 0 , $a.P$ et $+$

Pour commencer, **Basic CCS** : pas de $|$, de $\setminus L$, de $[f]$, de constantes de processus.

⇒ juste 0 , $a.P$ et $+$

$$(A1) \quad X + Y = Y + X$$

$$(A2) \quad X + (Y + Z) = (X + Y) + Z$$

$$(A3) \quad X + 0 = X$$

$$(A4) \quad X + X = X$$

$$a.(b.0 + (c.0 + b.0)) + 0 \stackrel{?}{=} a.(b.0 + c.0)$$

- A1 – A4 **corrects** pour \sim et Basic CCS
 - $P = Q$ prouvé $\Rightarrow P \sim Q$
 - pourquoi ? on peut construire des biss. fortes
 - ex: $\forall P \in \mathcal{P}, \{ \langle P + P, P \rangle \} \cup \sim$ est une bissimulation

- A1 – A4 **corrects** pour \sim et Basic CCS
 - $P = Q$ prouvé $\Rightarrow P \sim Q$
 - pourquoi ? on peut construire des biss. fortes
 - ex: $\forall P \in \mathcal{P}, \{ \langle P + P, P \rangle \} \cup \sim$ est une bissimulation

- A1 – A4 **complets** pour \sim et Basic CCS
 - $P \sim Q \Rightarrow P = Q$ prouvable
 - pourquoi ? si $P \xrightarrow{a} Q$ alors on peut prouver $P = a.Q + R$ pour un R

Basic Parallel CCS = Basic CCS + |

■ $A1 - A4$ corrects pour Basic Parallel CCS

⇒ il faut juste ajouter un axiome pour gérer | : la loi d'expansion

Notation :

I ensemble fini, $\{P_i | i \in I\}$ ensemble I -indexé de processus

$$\sum_{i \in I} P_i = \begin{cases} 0 & \text{si } I = \emptyset \\ P_{i_1} & \text{si } I = \{i_1\} \\ P_{i_1} + \dots + P_{i_n} & \text{si } I = \{i_1, \dots, i_n\} \text{ et } n \geq 2 \end{cases}$$

soit $P = \sum_{i \in I} a_i \cdot P_i$ et $Q = \sum_{j \in J} b_j \cdot Q_j$.

$$\begin{aligned}
 (EXP) \quad P|Q &= \sum_{i \in I} a_i \cdot (P_i|Q) \\
 &+ \sum_{j \in J} b_j \cdot (P|Q_j) \\
 &+ \sum_{\langle i, j \rangle \in \{\langle i, j \rangle \in I \times J \mid a_i = \bar{b}_j\}} \tau \cdot (P_i|Q_j)
 \end{aligned}$$

$$a.0 | (\bar{a}.0 + b.0) \stackrel{?}{=} a.(\bar{a}.0 + b.0) + \bar{a}.a.0 + b.a.0 + \tau.0$$

- $A1 - A4$, *EXP* corrects pour \sim et Basic Parallel CCS
 - pourquoi ? on peut construire des biss. fortes

- $A1 - A4$, *EXP* corrects pour \sim et Basic Parallel CCS
 - pourquoi ? on peut construire des biss. fortes
- $A1 - A4$, *EXP* complets pour \sim et Basic Parallel CCS
 - pourquoi ?
 - car *EXP* permet à $|$ d'être "mis à l'intérieur", evt. supprimé si les arguments ont la "bonne forme"
 - les arguments des occurrences internes de $|$ peuvent être mis en "bonne forme" en utilisant $A1 - A4$

Dernier fragment de CCS

- étend Basic Parallel CCS avec $\setminus L$ et $[f]$
- toujours pas de constantes de processus

- $A1 - A4, EXP$ corrects pour Finite CCS
 - ➡ il faut juste ajouter des axiomes pour gérer $\setminus L$ et $[f]$

$$(RES1) \quad 0 \setminus L = 0$$

$$(RES2) \quad (a.P) \setminus L = \begin{cases} 0 & \text{si } a \in L \text{ ou } \bar{a} \in L \\ a.(P \setminus L) & \text{sinon} \end{cases}$$

$$(RES3) \quad (P + Q) \setminus L = (P \setminus L) + (Q \setminus L)$$

$$(REL1) \quad 0[f] = 0$$

$$(REL2) \quad (a.P)[f] = \hat{f}(a).(P[f])$$

$$(REL3) \quad (P + Q)[f] = (P[f]) + (Q[f])$$

- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3$ corrects pour \sim et Finite CCS
 - pourquoi ? on peut construire des biss. fortes

- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3$ corrects pour \sim et Finite CCS
 - pourquoi ? on peut construire des biss. fortes
- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3$ complets pour \sim et Finite CCS
 - pourquoi ?
 - utiliser EXP pour supprimer occurrences de haut niveau de $|$ dans $\backslash L$ et $[f]$
 - utiliser alors RES_i et REL_i pour descendre $\backslash L$ et $[f]$ sous $a.P$ et $+$, puis les enlever

- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3$
axiomes corrects pour \approx^C (car $p \approx q \Rightarrow p \approx^C q$)
- permettent de réécrire un terme CCS en terme Basic CCS
 - ⇒ pour gérer \approx^C , il faut des axiomes pour gérer τ p/r aux opérateurs de Basic CCS

- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3$
axiomes corrects pour \approx^C (car $p \approx q \Rightarrow p \approx^C q$)
- permettent de réécrire un terme CCS en terme Basic CCS
- ⇒ pour gérer \approx^C , il faut des axiomes pour gérer τ p/r aux opérateurs de Basic CCS

$$(\tau 1) \quad a.\tau.P \quad = \quad a.P$$

$$(\tau 2) \quad P + \tau.P \quad = \quad \tau.P$$

$$(\tau 3) \quad a.(P + \tau.Q) \quad = \quad a.(P + \tau.Q) + a.Q$$

- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3$
axiomes corrects pour \approx^C (car $p \approx q \Rightarrow p \approx^C q$)
- permettent de réécrire un terme CCS en terme Basic CCS
- ➡ pour gérer \approx^C , il faut des axiomes pour gérer τ p/r aux opérateurs de Basic CCS

$$(\tau 1) \quad a.\tau.P \quad = \quad a.P$$

$$(\tau 2) \quad P + \tau.P \quad = \quad \tau.P$$

$$(\tau 3) \quad a.(P + \tau.Q) \quad = \quad a.(P + \tau.Q) + a.Q$$

Attention : $\tau.P = P$ n'est pas correct (permettrait de prouver $\tau.a.0 = a.0$ alors que $\tau.a.0 \not\approx^C a.0$)

- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3, \tau1 - \tau3$
corrects pour \approx^C et Finite CCS
 - pourquoi ? on peut construire des biss. faibles

- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3, \tau1 - \tau3$
corrects pour \approx^C et Finite CCS
 - pourquoi ? on peut construire des biss. faibles
- $A1 - A4, EXP, RES1 - RES3, REL1 - REL3, \tau1 - \tau3$
complets pour \approx^C et Finite CCS
 - pourquoi ? c'est magique :) (voir le HPA)

PAS POSSIBLE !

- CCS a le pouvoir d'expression des machines de Turing
- \sim , \approx^C , non récursivement énumérables, or l'ensemble des choses déductibles doivent l'être
 - ⇒ aucune axiomatisation complète possible de \sim ou \approx^C ne peut exister

PAS POSSIBLE !

- CCS a le pouvoir d'expression des machines de Turing
- \sim , \approx^C , non récursivement énumérables, or l'ensemble des choses déductibles doivent l'être
 - ⇒ aucune axiomatisation complète possible de \sim ou \approx^C ne peut exister

Que faire ?

- définir des règles d'inférence pour des classes restreintes de CCS
 - ⇒ suffit en général ...

- il existe beaucoup d'algèbres de processus
- toutes mettent l'accent sur sémantique opérationnelle et raisonnement équationnel (même si CSP est bcp + axé dénotationnel)
- autres approches : à base de (S)TS, logiques temporelles, Réseaux de Petri, UML