

PPC

Cours 06 - π -calcul

Romain Demangeon

11001 - M2 STL

11/10/2017

Pourquoi apprendre le π -calcul

π -calcul: calcul formel concurrent par passage de messages.

- ▶ On **ne programme pas** en π -calcul.
- ▶ Le π -calcul est un **modèle** des comportements des programmes concurrents, distribués, répartis.
 - ▶ il ne modélise que la partie **observable**: envoi et réception de messages.
- ▶ On s'en sert pour étudier des propriétés de la **concurrency** (bloquages, terminaison, fuite d'information)
 - ▶ Prouver des théorèmes sur des modèles permet:
 - ▶ de développer des nouveaux langages sûrs,
 - ▶ de découvrir des techniques de programmation,
 - ▶ de construire des **vérifications** de programmes réels.
 - ▶ Il y a parfois une **chaîne** entre la pratique et la théorie.
 π -calcul typé $\longrightarrow$ types de sessions $\longrightarrow$ *Scribble* $\longrightarrow$ Moniteurs (Python)
- ▶ Pourquoi l'apprendre en **M2 STL**:
 - ▶ pour **lire** des papiers sur la concurrence,
 - ▶ pour développer des **automatismes** de programmation,
 - ▶ pour comprendre la **difficulté** de la programmation distribuée,
 - ▶ pour la **culture** générale informatique.

- ▶ CCS est **Turing-complet**. (CSS aussi)
- ▶ Utilisation dans des **modèles** de programmes concurrents:
 - ▶ **passage de messages**: comment parler des messages ?
 $question.\overline{reponse}.0$
l'action $\overline{reponse}$ **transporte** une information.

- ▶ CCS est **Turing-complet**. (CSS aussi)
- ▶ Utilisation dans des **modèles** de programmes concurrents:
 - ▶ **passage de messages**: comment parler des messages ?
 $question.\overline{reponse}.0$
l'action $\overline{reponse}$ **transporte** une information.
 - ▶ messages dans **un ensemble fini**:
 $question.(\overline{reponse_true}.0 + \overline{reponse_false}.0)$
messages **paramétrés** (entiers) ?

- ▶ CCS est **Turing-complet**. (CSS aussi)
- ▶ Utilisation dans des **modèles** de programmes concurrents:
 - ▶ **passage de messages**: comment parler des messages ?
 $question.\overline{reponse}.0$
l'action $\overline{reponse}$ **transporte** une information.
 - ▶ messages dans **un ensemble fini**:
 $question.(\overline{reponse_true}.0 + \overline{reponse_false}.0)$
messages **paramétrés** (entiers) ?
- ▶ **Value-passing** CCS: $question.\overline{reponse}\langle 42 \rangle.0$

- ▶ CCS est **Turing-complet**. (CSS aussi)
- ▶ Utilisation dans des **modèles** de programmes concurrents:
 - ▶ **passage de messages**: comment parler des messages ?
 $question.\overline{reponse}.0$
l'action $\overline{reponse}$ **transporte** une information.
 - ▶ messages dans **un ensemble fini**:
 $question.(\overline{reponse_true}.0 + \overline{reponse_false}.0)$
messages **paramétrés** (entiers) ?
- ▶ **Value-passing** CCS: $question.\overline{reponse}\langle 42 \rangle.0$
- ▶ **Ordre supérieur**: $question(rep).\overline{rep}\langle 42 \rangle.0$
 - ▶ **π -calcul**: les **messages** envoyés sur les canaux sont des **noms**.

"Lettre-grecque"-calculs

- ▶ λ -calcul (1930): programmation fonctionnelle
 $M, N ::= \lambda x.M \mid M N \mid x$
- ▶ μ -calcul (1983): formules logiques "infinies":
 $\phi ::= \top \mid \perp \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \Rightarrow \phi \mid X \mid \langle \rangle \phi \mid [] \phi \mid \mu X.\phi \mid \nu X.\phi$
- ▶ π -calcul (1992): passage de messages et mobilité:
 $P, Q ::= 0 \mid a(x).P \mid \bar{a}\langle v \rangle.P \mid !a(x).P \mid (P \mid Q) \mid (\nu a) P \mid (P + Q)$
- ▶ σ -calcul (1995): programmation objet:
 $a, b ::= [l_j = \sigma(x_j)b_j] \mid a.l_j \mid a.l_j := \sigma(x)b$
- ▶ ρ -calcul (1998): formalisme pour la réécriture:
 $t ::= x \mid f(t, \dots, t) \mid \{t, \dots, t\} \mid u_{[E]} \rightarrow t \mid [t]t$
- ▶ d'autres variantes: $\lambda_{\sigma\uparrow}$ -calcul (substitutions explicites), $\lambda_{\mu\mu'}$ -calcul (pile d'exécutions), $\text{HO}\pi$ (processus dans les messages)

Syntaxe (préliminaires)

Noms

On dispose d'un ensemble infini de **noms**. $\mathcal{N} = \{a, b, c, \dots, x, y, \dots\}$

- ▶ Comme en **CCS**, les noms sont des **canaux**.

Substitutions

On note $P[x/y]$ le processus obtenu à partir de P en **remplaçant toutes les occurrences** du nom y par le nom x .

- ▶ Similaire à $M[N/x]$ en λ , mais plus **simple** (nom pour nom).
- ▶ **Exemple** en CCS: si $P = a.\bar{b} \mid \bar{b}.a$ alors $P[c/b] = a.\bar{c} \mid \bar{c}.a$ et $P[a/b] = a.\bar{a} \mid \bar{a}.a$.
 - ▶ une substitution peut changer la **sémantique**.
- ▶ Ici, pas de **variables**, tout est **nom**.

$P ::= 0 \mid \bar{a}\langle v \rangle.P \mid a(x).P \mid !P \mid (P \mid Q) \mid (\nu a) P \mid (P + Q)$

- ▶ deux différences depuis CCS: **passage de messages** et **réplication**.
- ▶ $\bar{a}\langle v \rangle.P$ le processus est **prêt à envoyer** un nom v sur le canal a , puis à continuer selon le processus P .
- ▶ $a(x).P$ le processus est **prêt à recevoir** un nom x sur le canal a , puis à continuer selon le processus P .
 - ▶ P peut **utiliser** x (notion de variable) !
- ▶ $!P$ est "une **infinité** de copie de P **en parallèle**"
 - ▶ modélise la récursion/**non-terminaison** $!(\bar{a}.0) \mid !(a.0)$
- ▶ On oublie les occurrences de 0 **derrière des préfixes** (comme en CCS):
 - ▶ $\bar{b}\langle c \rangle.c(x).0$ s'écrit $\bar{b}\langle c \rangle.c(x)$
 - ▶ $\bar{a}\langle v \rangle.(c(x).0 \mid !\bar{a}\langle c \rangle.0)$ s'écrit $\bar{a}\langle v \rangle.(c(x) \mid !\bar{a}\langle c \rangle)$
- ▶ Notion de **nom lié**:
 - ▶ $a(x).P$ **lie** x dans P ,
 - ▶ $(\nu a) P$ **lie** a dans P ,
 - ▶ **α -équivalence** de noms liés possible.
 - ▶ **Convention de Barendregt**: les noms liés sont distincts deux à deux et sont distinct des noms libres.

Congruence Structurale

$P \mid 0 \equiv P$	neutre
$P \mid Q \equiv Q \mid P$	commutativité
$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$	associativité
$P + 0 \equiv P$	neutre +
$P + Q \equiv Q + P$	commutativité +
$P + (Q + R) \equiv (P + Q) + R$	associativité +
$(\nu a) (\nu b) P \equiv (\nu b) (\nu a) P$	commutativité ν
$(\nu c) 0 \equiv 0$	neutre ν
$(\nu a) (P \mid Q) \equiv (\nu a) P \mid Q$	si $a \notin Q$ extrusion de portée
$!P \equiv !P \mid P$	réplication

$$\begin{array}{c}
 \frac{P \equiv P' \quad Q \equiv Q'}{P \mid Q \equiv P' \mid Q'} \\
 \frac{P \equiv P' \quad Q \equiv Q'}{P + Q \equiv P' + Q'} \\
 \frac{P \equiv P'}{(\nu a) P \equiv (\nu a) P'} \\
 \frac{P \equiv P'}{!P \equiv !P'} \\
 \frac{P \equiv P'}{a(x).P \equiv a(x).P'} \\
 \frac{P \equiv P'}{\bar{a}\langle v \rangle.P \equiv \bar{a}\langle v \rangle.P'}
 \end{array}$$

- ▶ axiomes: bases structurelles,
- ▶ règles: être une congruence.

- ▶ Comme en CCS, permet de dire quand deux processus sont égaux.
 - ▶ $(\nu \nu) (\bar{a}\langle v \rangle \mid !a(x) \mid \bar{a}\langle b \rangle) \equiv \bar{a}\langle b \rangle \mid a(x) \mid !a(x) \mid a(x) \mid (\nu \nu) (\bar{a}\langle v \rangle)$
- ▶ **Extrusion**: origine de la notion de mobilité.
- ▶ **Réplication**: $!P$ est inutilisable en l'état.
 - ▶ **dépliage** (*unfolding*).

$$\text{(Com)} \quad \frac{}{a(x).P \mid \bar{a}\langle v \rangle.Q \longrightarrow P[v/x] \mid Q}$$

$$\text{(Par)} \quad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$$

$$\text{(Res)} \quad \frac{P \longrightarrow P'}{(\nu a) P \longrightarrow (\nu a) P'}$$

$$\text{(Cong)} \quad \frac{Q \equiv P \quad P \longrightarrow P' \quad P' \equiv Q'}{Q \longrightarrow Q'}$$

- ▶ Similaire à CCS.
- ▶ **message**: "le message v est transmis à P'' ".
- ▶ **réplication**: n'apparaît pas dans la sémantique.
 - ▶ on utilise **(Cong)** pour la déplier.

$$\text{(Com)} \quad \frac{}{(a(x).P_1 + P_2) \mid (\bar{a}\langle v \rangle.Q_1 + Q_2) \longrightarrow P_1[v/x] \mid Q_1}$$

$$\text{(Par)} \quad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$$

$$\text{(Res)} \quad \frac{P \longrightarrow P'}{(\nu a) P \longrightarrow (\nu a) P'}$$

$$\text{(Cong)} \quad \frac{Q \equiv P \quad P \longrightarrow P' \quad P' \equiv Q'}{Q \longrightarrow Q'}$$

- ▶ Similaire à CCS.
- ▶ **message**: "le message v est transmis à P_1 ".
- ▶ **réplication**: n'apparaît pas dans la sémantique.
 - ▶ on utilise **(Cong)** pour la déplier.
- ▶ Utilisation du **choix**.

Sémantique de réd. (exemples)

- ▶ **Concurrence:** $\bar{a}\langle b \rangle \mid \bar{a}\langle c \rangle \mid a(x).\bar{d}\langle x \rangle$
- ▶ **Choix:** $\bar{a}\langle b \rangle + \bar{a}\langle c \rangle \mid a(x).\bar{d}\langle x \rangle$
- ▶ **Concurrence et blocage:** $\bar{a}\langle b \rangle \mid \bar{a}\langle c \rangle \mid a(x).\bar{x}\langle v \rangle \mid b(y).a(z)$
- ▶ **Comportement infini:** $!\bar{a}\langle v \rangle.a(x) \mid a(x)$
- ▶ **Comportement infini, infinité de noms:** $!(\nu v) (\bar{a}\langle v \rangle.a(x)) \mid a(x)$
- ▶ **Comportement infini, évitable:** $a(y) \mid !a(x).\bar{a}\langle x \rangle \mid \bar{a}\langle v \rangle$
- ▶ **Comportement infini, évitable:** $c(y) \mid c(z).!z(x).\bar{a}\langle x \rangle \mid \bar{a}\langle v \rangle \mid \bar{c}\langle a \rangle$
- ▶ **Comportement infini, type récursif:** $!a(x).\bar{x}\langle a \rangle \mid (\bar{a}\langle b \rangle + \bar{a}\langle a \rangle)$

Syntaxe des préfixes polyadiques

Le π -calcul polyadique utilise les préfixes $a(x_1, x_2, \dots, x_n).P$ et $\bar{a}\langle v_1, v_2, \dots, v_n \rangle.Q$ avec $n \in \mathbb{N}$.

- **Idée**: transporter **plusieurs** messages en même temps.

- **Sémantique**:

$$a(x_1, \dots, x_n).P \mid \bar{a}\langle v_1, \dots, v_n \rangle \longrightarrow P[v_1, \dots, v_n/x_1, \dots, x_n] \mid Q$$

- **Convention de Barendregt**:

$$P[v_1, \dots, v_n/x_1, \dots, x_n] = P[v_1/x_1] \dots [v_n/x_n]$$

- $n = 0$: pas de messages

- on note a pour $a()$,
- on note $\bar{a}$ pour $\bar{a}\langle \rangle$,
- on retrouve **CCS** !

- **Exemple**: $a(x, y).(\bar{x} \mid y) \mid (\bar{a}\langle b, c \rangle + \bar{a}\langle b, b \rangle)$

Définition

La **Mobilité** est la capacité de modéliser des systèmes dont la **topologie évolue à l'exécution**.

- ▶ Utilisation de la *scope extrusion*.
- ▶ $P_1 = (\nu b) (\bar{a}\langle b \rangle)$ et $P_2 = !c(r).\bar{r} \mid a(x).!x(r).\bar{r}$ dans $P = (\nu a) (P_1 \mid P_2)$
 - ▶ initialement, P ne peut recevoir des requêtes **que sur c** (car P_2 ne connaît pas b).
 - ▶ après réductions, P peut recevoir des requêtes **sur b** .
- ▶ On peut **créer**, **transporter** et **utiliser ailleurs** des noms.
- ▶ les **liens** entre process mis en parallèle (noms **en communs**) évoluent à l'exécution.
- ▶ Trait **caractéristique** de π . *A Calculus of Mobile Processes*, Milner, Parrow, Walker.

Réception répliquée

- ▶ Pour l'efficacité, la terminaison, la modélisation: utilité d'une notion de taille (nombre de préfixes).
- ▶ Un processus avec réplication n'a pas de taille:
$$(!\bar{a}\langle v \rangle \mid a(x)) \equiv (!\bar{a}\langle v \rangle \mid \bar{a}\langle v \rangle \mid a(x)) \equiv (!\bar{a}\langle v \rangle \mid \bar{a}\langle v \rangle \mid \bar{a}\langle v \rangle \mid a(x)) \equiv \dots$$
- ▶ Malaise:
 - ▶ Comment se représenter un tel processus ?
 - ▶ Qu'est ce qu'on cherche à modéliser ?

Réception répliquée

- ▶ Pour l'efficacité, la terminaison, la modélisation: utilité d'une notion de taille (nombre de préfixes).
- ▶ Un processus avec réplication n'a pas de taille:
 $(! \bar{a}\langle v \rangle \mid a(x)) \equiv (! \bar{a}\langle v \rangle \mid \bar{a}\langle v \rangle \mid a(x)) \equiv (! \bar{a}\langle v \rangle \mid \bar{a}\langle v \rangle \mid \bar{a}\langle v \rangle \mid a(x)) \equiv \dots$
- ▶ Malaise:
 - ▶ Comment se représenter un tel processus ?
 - ▶ Qu'est ce qu'on cherche à modéliser ?

Solution

Voire la réplication de manière réactive.

- ▶ Ne considérer que la réception répliquée $!a(x).P$,
- ▶ Interdire le dépliage $!P \equiv P \mid !P$,
- ▶ Ajouter une règle à la sémantique.

$$\frac{}{!a(x).P \mid \bar{a}\langle v \rangle.Q \longrightarrow !a(x).P \mid P[v/x] \mid Q}$$

Réception répliquée (II)

- ▶ π avec réception répliquée a la même expressivité que π avec la réplication.
 - ▶ on ne perd rien à se restreindre à la réception répliquée.
- ▶ Modèle du serveur, toujours disponible, qui lance des calculs:
$$\begin{aligned}S &= !url(arg, can).[...].\overline{can}\langle res \rangle \\C_1 &= (\nu c_1) \overline{url}\langle 18, c_1 \rangle.c_1(x).[...] \\C_2 &= (\nu c_2) \overline{url}\langle 42, c_2 \rangle.c_2(y).[...]\end{aligned}$$
- ▶ On peut définir la taille comme (par exemple) le nombre de préfixes libres (qui ne sont pas sous des !).
 - ▶ Divergence, taille constante: $!a(x).\bar{a}\langle x \rangle \mid \bar{a}\langle v \rangle$
 - ▶ Divergence, taille explosive: $!a(x).(\bar{a}\langle x \rangle \mid \bar{a}\langle x \rangle) \mid \bar{a}\langle v \rangle$

Objectif: trouver une bonne **équivalence comportementale**.

Définition

Une relation $\mathcal{R}$ entre processi est une **bisimulation de réduction** si pour tous P et Q tels que $P\mathcal{R}Q$:

- ▶ pour tout P' t.q. $P \longrightarrow P'$, il existe Q' t.q. $Q \longrightarrow Q'$ **et** $P'\mathcal{R}Q'$
- ▶ pour tout Q' t.q. $Q \longrightarrow Q'$, il existe P' t.q. $P \longrightarrow P'$ **et** $P'\mathcal{R}Q'$

La **bisimilarité de réduction** est la plus grande bisimulation.

- ▶ Cette relation n'est pas très **discriminante**:
 - ▶ 0 et $a(x).(!b \mid \bar{b}.((\nu d) \bar{c}\langle d \rangle))$ sont **bisimilaires de réduction**.

Objectif: trouver une bonne **équivalence comportementale**.

Définition

Une relation $\mathcal{R}$ entre processi est une **bisimulation de réduction** si pour tous P et Q tels que $P\mathcal{R}Q$:

- ▶ pour tout P' t.q. $P \longrightarrow P'$, il existe Q' t.q. $Q \longrightarrow Q'$ **et** $P'\mathcal{R}Q'$
- ▶ pour tout Q' t.q. $Q \longrightarrow Q'$, il existe P' t.q. $P \longrightarrow P'$ **et** $P'\mathcal{R}Q'$

La **bisimilarité de réduction** est la plus grande bisimulation.

- ▶ Cette relation n'est pas très **discriminante**:
 - ▶ 0 et $a(x).(!b \mid \bar{b}.((\nu d) \bar{c}\langle d \rangle))$ sont **bisimilaires de réduction**.
 - ▶ C'est **nul** !
- ▶ Pour obtenir une équivalence intéressante, on va s'intéresser aux **transitions**.

Sémantiques de transition

Etiquettes: $\alpha ::= ax \mid \bar{a}b$

$$\text{(Out)} \quad \frac{}{\bar{a}\langle v \rangle . P \xrightarrow{\bar{a}v} P}$$

$$\text{(In)} \quad \frac{}{a(x) . P \xrightarrow{av} P[v/x]}$$

$$\text{(Comm)} \quad \frac{P \xrightarrow{av} P' \quad Q \xrightarrow{\bar{a}v} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\text{(Par)} \quad \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

$$\text{(Sum)} \quad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

$$\text{(Res)} \quad \frac{P \xrightarrow{\alpha} P' \quad a \notin \alpha}{(\nu a) P \xrightarrow{\alpha} (\nu a) P'}$$

(+ règles symétriques.)

- Pas de **congruence structurelle**.

Sémantiques de transition

Etiquettes: $\alpha ::= ax \mid \bar{a}b \mid \bar{a}(b)$

(Out)

$$\frac{}{\bar{a}\langle v \rangle . P \xrightarrow{\bar{a}v} P}$$

(In)

$$\frac{}{a(x) . P \xrightarrow{av} P[v/x]}$$

(Comm)

$$\frac{P \xrightarrow{av} P' \quad Q \xrightarrow{\bar{a}v} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

(Par)

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

(Sum)

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

(Res)

$$\frac{P \xrightarrow{\alpha} P' \quad a \notin \alpha}{(\nu a) P \xrightarrow{\alpha} (\nu a) P'}$$

(Open)

$$\frac{P \xrightarrow{\bar{a}b} P'}{(\nu b) P \xrightarrow{\bar{a}(b)} P'}$$

(Close)

$$\frac{P \xrightarrow{ab} P' \quad Q \xrightarrow{\bar{a}(b)} Q'}{P \mid Q \xrightarrow{\tau} (\nu b) (P' \mid Q')}$$

(Rep)

$$\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P}$$

(RepCom)

$$\frac{P \xrightarrow{ab} P' \quad P \xrightarrow{\bar{a}b} P''}{!P \xrightarrow{\tau} (P' \mid P'') \mid !P}$$

(RepClo)

$$\frac{P \xrightarrow{ab} P' \quad P \xrightarrow{\bar{a}(b)} P''}{!P \xrightarrow{\tau} (\nu b) (P' \mid P'') \mid !P}$$

(+ règles symétriques.)

- Pas de **congruence structurelle**.
- Défi: **extrusion** de portée et **réplication**.
- **exemple**: $P_1 = (\nu b) (\bar{a}\langle b \rangle)$, $P_2 = a(x).(\bar{x} \mid x)$ et $P = (\nu a) (P_1 \mid P_2)$

Harmonie

- ▶ Si $P \equiv \xrightarrow{\alpha} P'$ alors $P \xrightarrow{\alpha} \equiv P'$
- ▶ Si $P \longrightarrow P'$ alors $P \xrightarrow{\tau} \equiv P'$

Définition

Une relation $\mathcal{R}$ entre processi est une bisimulation (forte) si pour tous P et Q tels que $P\mathcal{R}Q$:

- ▶ pour tout (P', α) t.q. $P \xrightarrow{\alpha} P'$, il existe Q' t.q. $Q \xrightarrow{\alpha} Q'$ **et** $P'\mathcal{R}Q'$
- ▶ pour tout (Q', α) t.q. $Q \xrightarrow{\alpha} Q'$, il existe P' t.q. $P \xrightarrow{\alpha} P'$ **et** $P'\mathcal{R}Q'$

On note $\sim$, la **bisimilarité** (forte), la plus grande relation $\mathcal{R}$.

- ▶ Relation plus **fine** que la bisimulation de réduction.
- ▶ Que faire si on veut une équivalence de **test** ?

Barbes

- ▶ $P \downarrow_a$ signifie " P peut recevoir un message sur a .
- ▶ $P \downarrow_{\bar{a}}$ signifie " P peut envoyer un message sur a (on ne dit pas quoi) .
- ▶ on note μ pour a ou $\bar{a}$ (mais pas τ)

Définition

Une relation $\mathcal{R}$ entre processi est une bisimulation barbue si pour tous P et Q tels que $P\mathcal{R}Q$:

- ▶ $\forall \mu$, si $P \downarrow_\mu$ alors $Q \downarrow_\mu$.
- ▶ $\forall \mu$, si $Q \downarrow_\mu$ alors $P \downarrow_\mu$.
- ▶ pour tout P' t.q. $P \xrightarrow{\tau} P'$, il existe Q' t.q. $Q \xrightarrow{\tau} Q'$ et $P'\mathcal{R}Q'$.
- ▶ pour tout Q' t.q. $Q \xrightarrow{\tau} Q'$, il existe P' t.q. $P \xrightarrow{\tau} P'$ et $P'\mathcal{R}Q'$.

- ▶ Que penser de $P_1 = a(x).\bar{x} \mid \bar{b} \mid b$ et $P_2 = a(y) \mid \bar{b} \mid b$?

Barbes

- ▶ $P \downarrow_a$ signifie " P peut recevoir un message sur a .
- ▶ $P \downarrow_{\bar{a}}$ signifie " P peut envoyer un message sur a (on ne dit pas quoi) .
- ▶ on note μ pour a ou $\bar{a}$ (mais pas τ)

Définition

Une relation $\mathcal{R}$ entre processi est une bisimulation barbue si pour tous P et Q tels que $P\mathcal{R}Q$:

- ▶ $\forall \mu$, si $P \downarrow_\mu$ alors $Q \downarrow_\mu$.
- ▶ $\forall \mu$, si $Q \downarrow_\mu$ alors $P \downarrow_\mu$.
- ▶ pour tout P' t.q. $P \xrightarrow{\tau} P'$, il existe Q' t.q. $Q \xrightarrow{\tau} Q'$ et $P'\mathcal{R}Q'$.
- ▶ pour tout Q' t.q. $Q \xrightarrow{\tau} Q'$, il existe P' t.q. $P \xrightarrow{\tau} P'$ et $P'\mathcal{R}Q'$.

- ▶ Que penser de $P_1 = a(x).\bar{x} \mid \bar{b} \mid b$ et $P_2 = a(y) \mid \bar{b} \mid b$?
- ▶ On ne "teste" pas assez.

Définition

L'équivalence barbue $\simeq$ est définie par:

- ▶ $P \simeq Q$ si $\forall R, P \mid R$ est bisimilaire barbu à $Q \mid R$.
- ▶ Que penser de $P_1 = a(x).\bar{x} \mid \bar{b} \mid b$ et $P_2 = a(y) \mid \bar{b} \mid b$?
- ▶ Que penser de $P_1 = a(x).\bar{x} \mid \bar{b} \mid b$ et $P_2 = a(y).\bar{c} \mid \bar{b} \mid b$?

Théorème de Caractérisation

$P \simeq Q$ si et seulement si $P \sim Q$.

- ▶ On capture la bisimilarité (une équivalence de transition) avec une équivalence de test.

- ▶ On a utilisé les règles de **transition**

$$\begin{array}{c}
 \text{(In)} \quad \frac{}{a(x).P \xrightarrow{av} P[v/x]} \\
 \text{(Comm)} \quad \frac{P \xrightarrow{av} P' \quad Q \xrightarrow{\bar{a}v} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}
 \end{array}$$

- ▶ On aurait pu prendre

$$\begin{array}{c}
 \text{(In)} \quad \frac{}{a(x).P \xrightarrow{a(x)} P} \\
 \text{(Comm)} \quad \frac{P \xrightarrow{a(x)} P' \quad Q \xrightarrow{\bar{a}v} Q'}{P \mid Q \xrightarrow{\tau} P'[v/x] \mid Q'}
 \end{array}$$

- ▶ Réception **précoce** vs. réception **tardive**.
- ▶ On récupère une nouvelle équivalence: **bisimilarité tardive** $\sim_I$:
 - ▶ pour tout $(P', \alpha \neq \text{réception})$ t.q. $P \xrightarrow{\alpha} P'$, il existe Q' t.q. $Q \xrightarrow{\alpha} Q'$ **et** $P' \mathcal{R} Q'$ (et vice-versa)
 - ▶ pour tout (P', a, x, v) t.q. $P \xrightarrow{a(x)} P'$, il existe Q' t.q. $Q \xrightarrow{a(x)} Q'$ **et** $P'[v/x] \mathcal{R} Q'[v/x]$ (et vice-versa)
- ▶ $P \sim_I Q \Rightarrow P \sim Q$.
- ▶ considérer $P_1 = a(x).(x \mid b)$ et $P_2 = a(x).(x.b + b.x) + a(x).(\nu c)((x.b + b.x + c) \mid \bar{c})$

Equivalences faibles

- ▶ Comme en CCS, on définit $\Rightarrow^\alpha$ comme $\xrightarrow{\tau} \dots \xrightarrow{\tau} \xrightarrow{\alpha} \xrightarrow{\tau} \dots \xrightarrow{\tau}$
- ▶ On peut définir une version **faible** de toutes les équivalences précédentes.
- ▶ Souvent plus utile en **modélisation**: on oublie les calculs internes.

Axiomatisation

- ▶ **Idée**: trouver des **formules logiques** qui décident quand deux termes sont **bisimilaires**.
 - ▶ pas besoin de faire le jeu infini.
- ▶ Des **axiomes**, des **régles** et des **schémas**.
- ▶ $P = P$, $P = Q \Rightarrow Q = P$, $P = Q \Rightarrow a(x).P = a(x).Q$, ...

Le π -calcul asynchrone est donné par:

$$P ::= 0 \mid \bar{a}\langle v \rangle \mid a(x).P \mid !P \mid (P \mid Q) \mid (\nu a) P \mid (P + Q)$$

(les sommes ne sont autorisées qu'avec des réceptions)

- ▶ **Idée:** les émissions ne sont pas bloquantes.
- ▶ **Idée:** une émission à top-level est considérée comme "partie".
- ▶ Plus proche d'un réseau.
- ▶ Comparer $a(x).\bar{x}.b.a(y).P$ et $a(x).(\bar{x} \mid \bar{b} \mid a(y).P)$.
- ▶ Les émissions peuvent être ordonnées tout de même:
 $(\nu b, c)(\bar{a}\langle 3, b \rangle \mid \bar{b}\langle 12, c \rangle \mid \bar{c}\langle \text{ok}' \rangle \mid P)$
- ▶ **Expressivité:** on peut encoder les communications synchrones.
 - ▶ $\bar{a}\langle v \rangle.Q$ devient $(\nu c)(\bar{a}\langle v, c \rangle \mid c.Q)$
 - ▶ $a(x).P$ devient $a(x, y).(\bar{y} \mid P)$

Référence: *The π -calculus A Theory of Mobile Processes*, Davide Sangiorgi, David Walker

- ▶ le π -calcul a une **action** τ .
 - ▶ **par exemple**: $a(x).(x.b + b.x + \tau)$
 - ▶ on peut s'en sortir **sans**.
- ▶ le π -calcul a une **comparaison de noms**.
 - ▶ **par exemple**: $a(x).a(y).([x = y]\bar{b} + \bar{c})$
 - ▶ on peut s'en sortir **sans**.
- ▶ Introduction de la **récursion**:
 - ▶ remplace la **réplication**,
 - ▶ $K = (x).P$ et $K[v]$
 - ▶ Par exemple $K = (x, y, z, t).\bar{x}\langle y \rangle.\bar{z}\langle t \rangle.K[z, t, x, y]$.
- ▶ Etude de la **congruence barbue**:
 - ▶ **contexte**: un terme avec un trou $C[]$.
 - ▶ **congruence barbue**: pour tout contexte $C[]$, $C[P]$ est **bisimilaire barbu** $C[Q]$.
 - ▶ Différence avec l'**équivalence**, le **préfixe** de réception $P_1 = \bar{x} \mid b$ et $P_2 = \bar{x}.b + b.\bar{x}$.
 - ▶ caractérise la bisimilarité **complète** $\forall \sigma. P\sigma \sim Q\sigma$.

$$P ::= 0 \mid \bar{a}\langle v \rangle.P \mid a\langle v \rangle.P \mid !P \mid (P \mid Q) \mid (\nu a) P \mid (P + Q)$$

- ▶ **Idée**: étudier la notion de **substitution**.
- ▶ Les noms des réceptions sont **libres**.
- ▶ **réduction**: $(\nu c) (\bar{a}\langle c \rangle.Q \mid a\langle v \rangle.P \mid R) \longrightarrow (P \mid Q \mid R)[v/c]$

$$P ::= 0 \mid \bar{a}\langle v \rangle.P \mid a\langle v \rangle.P \mid !P \mid (P \mid Q) \mid (\nu a) P \mid (P + Q) \mid [x = y]$$

- ▶ Fusions **explicites**.
- ▶ Les égalités apparaissent dans le calcul.
- ▶ **réduction**: $\bar{a}\langle c \rangle.Q \mid a\langle v \rangle.P \longrightarrow P \mid Q \mid [v = c]$
- ▶ **réécriture**: $P \mid [a = b] \equiv P[b/a] \mid [a = b]$

$$v, w ::= () \mid (x).P \mid v[w]$$

$$P ::= 0 \mid a(v).P \mid \bar{a}\langle v \rangle.P \mid (P \mid Q) \mid (P + Q) \mid (\nu c) P$$

- ▶ **Idée**: plutôt que d'envoyer des canaux, on envoie des **fonctions** qui prennent un paramètre et **renvoient un processus**.
- ▶ $()$ c'est l'unité de type $\diamond$.
- ▶ $(x).P$ est une **abstraction**, la fonction $x \mapsto P$ de type $T \longrightarrow \diamond$
- ▶ $v[w]$ est une **application**, la valeur v appliquée à w .

$$\bar{a}\langle (x).\bar{b}\langle x \rangle \rangle \mid b(y).y[()] \mid c(X) \mid \bar{a}\langle (z).\bar{c}\langle () \rangle \rangle$$

- ▶ On peut se restreindre au **fonction** de type $\star \longrightarrow \diamond$.
 - ▶ fonctions d'**ordre 1**.
 - ▶ c'est à dire à dire les **processi**.
- ▶ On passe les **processus** eux-mêmes sur des **canaux**.
- ▶ **exemple**: $\bar{a}\langle b.\bar{b} \rangle \mid a(X).(X \mid X \mid \bar{b}.b)$
- ▶ **divergence** sans réplication $P_0 = a(X).(X \mid \bar{a}\langle X \rangle)$ dans $\bar{a}\langle P_0 \rangle \mid P_0$
- ▶ **encodage** de HOp_i₁ dans π
 - ▶ $\bar{a}\langle R \rangle.Q$ devient $(\nu c) (\bar{a}\langle c \rangle.Q' \mid !c.R')$ (transfo. appliquée à Q et R)
 - ▶ $a(X).P$ devient $a(x).P'$ (transfo. appliquée à P)
 - ▶ X devient $\bar{x}$
- ▶ l'inverse **existe** aussi.

- ▶ $M, N ::= \lambda x.M \mid M N \mid x$
 - ▶ **Objectif**: modéliser la programmation fonctionnelle.
 - ▶ β -réduction: $\lambda x.M N \longrightarrow M[N/x]$
-
- ▶ Encodage de λ (stratégie CbV) dans π .
 - ▶ Encodage paramétré par un canal de retour (passage par les **continuations**).
 - ▶ $[\lambda x.M]_p = (\nu v) \bar{p}\langle v \rangle . !v(x, q)[M]_q$
 - ▶ $[x]_p = \bar{p}\langle x \rangle$
 - ▶ $[M N]_p = (\nu n, m) ([M]_m \mid [N]_n \mid m(x).n(y).\bar{x}\langle y, p \rangle)$

λ dans π , exemple

- Soit le **terme** $II = (\lambda x.x) (\lambda y.y)$.

- Son **encodage** sur p_1 est:

$$P_{II} = (\nu q_1, r_1) (\nu y_2) (\overline{q_1}\langle y_2 \rangle. !y_2(x, q_2). \overline{q_2}\langle x \rangle) \mid (\nu y_3) (\overline{r_1}\langle y_3 \rangle. !y_3(y, q_3). \overline{q_3}\langle y \rangle) \\ \mid q_1(y_1). r_1(z_1). \overline{y_1}\langle z_1, p_1 \rangle)$$

- Il se **réduit** (administrativement):

$$P_{II} \longrightarrow \longrightarrow (\nu q_1, r_1, y_2, y_3) (!y_2(x, q_2). \overline{q_2}\langle x \rangle) \mid (!y_3(y, q_3). \overline{q_3}\langle y \rangle) \mid \overline{y_2}\langle y_3, p_1 \rangle)$$

- L'application a **lieu**:

$$P_{II} \longrightarrow \longrightarrow \longrightarrow (\nu q_1, r_1, y_2, y_3) (!y_2(x, q_2). \overline{q_2}\langle x \rangle) \mid (!y_3(y, q_3). \overline{q_3}\langle y \rangle) \mid \overline{p_1}\langle y_3 \rangle)$$

- A **comparer** avec:

$$[\lambda y.y]_{p_1} = (\nu y_3) \overline{p_1}\langle y_3 \rangle. !y_3(y, q_3). \overline{q_3}\langle y \rangle$$

La prochaine fois:

- ▶ π -calculs typé,
- ▶ application: types de sessions.