PPC
Cours 7 - Vérification des Systèmes Concurrents: π-calcul typé et types de session

Romain Demangeon

5I872 (PPC) - M2 STL

2022

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

HOpi (Rappels du cours 06)

$v, w ::= () \mid (x).P \mid v\lfloor w \rfloor$
$P ::= 0 \mid a(v).P \mid \overline{a}\langle v \rangle.P \mid (P \mid Q) \mid (P + Q) \mid (\nu c)\ P$

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

$\overline{a}\langle (x).\overline{b}\langle x \rangle \rangle \mid b(y).y\lfloor () \rfloor \mid c(X) \mid \overline{a}\langle (z).\overline{c}\langle () \rangle \rangle$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

HOpi_1 (Rappels du cours 06)

- On peut se restreindre au fonction de type $\bigstar \longrightarrow \diamond$.
  - fonctions d'ordre 1.
  - c'est à dire à dire les processi.
- On passe les processus eux-mêmes sur des canaux.
- exemple: $\overline{a}\langle b.\overline{b}\rangle \mid a(X).(X \mid X \mid \overline{b}.b)$
- divergence sans réplication $P_0 = a(X).(X \mid \overline{a}\langle X\rangle)$ dans $\overline{a}\langle P_0\rangle \mid P_0$
- encodage de HOpi_1 dans π
  - $\overline{a}\langle R\rangle.Q$ devient $(\nu c)\ (\overline{a}\langle c\rangle.Q' \mid !c.R')$ (transfo. appliquée à Q et R)
  - $a(X).P$ devient $a(x).P'$ (transfo. appliquée à P)
  - X devient $\overline{x}$
- l'inverse existe aussi.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Passivation

$$P, Q, R \quad ::= \quad a(x).P \mid !a(x).P \mid \overline{a}\langle v\rangle.Q \mid (\nu a)\ P \mid (P \mid P) \mid P + Q \\ \mid \overline{a}\langle P\rangle.Q \mid a(X).P \mid !a(X).P \mid X \\ \mid \lfloor P \rfloor_a \mid a(X) \triangleright P$$

- Calcul pour les gouverner tous:
  - π-calcul avec réplication,
  - HOpi_1 (avec réplication),
  - Calcul localisé avec passivation.
- Sémantique:
  - $\lfloor R \rfloor_a \mid a(X) \triangleright P \longrightarrow P[R/X]$.
  - $R \longrightarrow R'$ implique $\lfloor R \rfloor_a \longrightarrow \lfloor R \rfloor_a$.
- Exporter du code: $\lfloor R \rfloor_l \mid l(X) \triangleright (\lfloor X \rfloor_l \mid \overline{a}\langle X\rangle)$
- Mettre à jour un processus: $\lfloor R \rfloor_l \mid a(X).l(Y) \triangleright \lfloor X \rfloor_l$
- Changer un processus d'endroit: $\lfloor R \rfloor_l \mid l(X) \triangleright \lfloor X \rfloor_{l'}$
- Utilisé dans la modélisation de systèmes distribués basés sur les composants (calcul Kell).

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

λ dans π

le λ-calcul

- $M, N ::= \lambda x.M \mid M\ N \mid x$
- 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).
- $[\lambda x.M]_p = (\nu v)\ \overline{p}\langle v\rangle.!v(x,q)[M]_q$
- $[x]_p = (\nu v)\ \overline{p}\langle x\rangle$
- $[M\ N]_p = (\nu n, m)\ ([M]_m \mid [N]_n \mid m(x).n(y).\overline{x}\langle y, p\rangle)$
- L'encodage se comporte comme le terme initial: Si $M \longrightarrow M'$ alors $[M]_p \Rightarrow P$ et $P \simeq [M']_p$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

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:

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

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Vérification: Motivation

- Objectif: Vérifier des systèmes concurrents par passage de message.
  - protocoles, programmes distribués, hardware, application web, ...
- Méthode:
  1. Abstraire les systèmes dans un langage mathématique.
  2. Prouver des propriétés sur l'abstraction.
  3. Vérifier les propriétés sur le système.

Abstraction des système concurrents

- CCS: les processus se synchronisent sur des canaux.
- CCS avec valeur: les processus communiquent des valeurs de bases (entiers, données)
- π-calcul: les processus communiquent des noms de canaux
  - Mobilité: la configuration du système change dynamiquement.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Vérification: Motivation (II)

Prouver des propriétés

- Quelles propriétés ?
  - terminaison, sûreté, sécurité, etc
- Types de propriétés:
  - syntaxiques (sur le processus)
  - sémantiques (sur ses réductions)
    - *safety*: "rien de mal n'arrive"
    - *liveness*: "une chose bonne finit par arriver"
  - comportementales (sur son équivalence vis à vis d'autres processus)

Vérifier les propriétés

- Statiquement, sur le code des programmes,
  - analyse statique,
  - typecheckers.
- Dynamiquement, à l'exécution,
  - avec des moniteurs.

Systèmes de Types

- Types: abstractions des termes, information sur leur nature.
  - exemple: types fonctionnels

    `f: list[int] * (int -> bool) -> list[int].`
- Objectifs:
  - garantir la saine composition des éléments d'un langage,
  - garantir d'autres propriétés (terminaison, *deadlock-freedom*, sécurité, bon comportement, ...)
- Méthode: définir des règles de typage qui prouvent des jugements de type $\Gamma \vdash P : T$.

Cas du π-calcul

- Quoi typer ?
  - les canaux.
- les règles correspondent à la structure de P:

$$\frac{\Gamma \vdash_\pi P_1 \qquad \Gamma \vdash_\pi P_2 \qquad \text{conditions sur } P_1 \text{ et } P_2}{\Gamma \vdash_\pi P_1 \mid P_2}$$

- Fixer l'usage d'un canal pour tous les utilisateurs.

Types Simples

- Principe: on donne comme type à un canal l'utilisation qui doit en être faite.
- Syntaxe des types: $T ::= Nat \mid Bool \mid \sharp\,(T, \ldots, T)$

- $p(x, y).(\overline{x}\langle y\rangle \mid \overline{y}\langle 3\rangle)$ typable avec $y : \sharp\, Nat$, $x : \sharp\,(\sharp\, Nat)$ et $p : \sharp\,(\sharp\,(\sharp\, Nat), \sharp\, Nat)$
- $\overline{a}\langle a\rangle$ pas typable (nécessite un type récursif)

Règle de l'output et du parallèle

$$\frac{\Gamma \vdash P \qquad \Gamma(a) = \sharp\, T \qquad \Gamma(v) = T}{\Gamma \vdash \overline{a}\langle v\rangle.P} \qquad \frac{\Gamma \vdash P_1 \qquad \Gamma \vdash P_2}{\Gamma \vdash P_1 \mid P_2}$$

Règle des types simples

$$(\textbf{Nil})\frac{}{\Gamma \vdash 0} \qquad (\textbf{Out})\frac{\Gamma \vdash P \qquad \Gamma(a) = \sharp\, T \qquad \Gamma(v) = T}{\Gamma \vdash \overline{a}\langle v \rangle.P}$$

$$(\textbf{Par})\frac{\Gamma \vdash P_1 \qquad \Gamma \vdash P_2}{\Gamma \vdash P_1 \mid P_2} \qquad (\textbf{In})\frac{\Gamma, x : T \vdash P \qquad \Gamma(a) = \sharp\, T}{\Gamma \vdash a(x).P}$$

$$(\textbf{Rep})\frac{\Gamma, x : T \vdash P \qquad \Gamma(a) = \sharp\, T}{\Gamma \vdash !a(x).P} \qquad (\textbf{Res})\frac{\Gamma, a : T \vdash P}{\Gamma \vdash (\nu a)\ P}$$

- s'étend au types simples polyadiques:
  - $(\textbf{Out})\dfrac{\Gamma \vdash P \qquad \Gamma(a) = \sharp\, T_1, \ldots, T_n \qquad \forall i, \Gamma(v_i) = T_i}{\Gamma \vdash \overline{a}\langle v_1, \ldots, v_n \rangle.P}$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Terminaison

Définition

Un programme termine quand toutes ses exécutions sont finies.

Programmes non-terminants (divergents)

- $(\lambda x.x\ x)\ (\lambda y.y\ y)$.
- while **true** do skip.
- $\{[\circ] \to [\circ\circ], [\circ] \to []\}$,

Terminaison en programmation

- pour les langages fonctionnels (systèmes de types, réalisabilité, λ_{ST}),
- pour les langages impératifs (interprétation abstraites, boucles),
- pour les systèmes de réécriture (ordres, interprétations poly.).

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Cas Particulier: λ-calcul simplement typé

- Abstraction des programmes fonctionnels séquentiels.
- Types simples ($A \to B$) garantissent la terminaison.
- Différentes preuves de terminaison.
- Peut être enrichi (polymorphisme, types sommes, ...).

En π, les types simples ne garantissent pas la terminaison:

- $!a(x).\overline{a}\langle x\rangle \mid \overline{a}\langle 3\rangle$ typable avec $a : \sharp\, Nat$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS
DEPUIS 1257

Terminaison des systèmes concurrents

Propriété intéressante

- Persistance des serveurs,
  mais on doit produire une réponse au bout d'un temps fini.
- Prérequis d'autres propriétés: *lock-freedom*, complexité, *fairness*.

Une Analyse Difficile

- Topologie dynamique, non-determinisme.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Divergence en π-calcul

- $D_1 = !a.\overline{a} \mid \overline{a}$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a}$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b \rangle \mid \overline{c}\langle a \rangle$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Divergence en π-calcul

- $D_1 = !a.\overline{a} \mid \overline{a} \longrightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a}$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b \rangle \mid \overline{c}\langle a \rangle$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Divergence en π-calcul

- $D_1 = !a.\overline{a} \mid \overline{a} \longrightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a} \longrightarrow\longrightarrow D_2$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b\rangle \mid \overline{c}\langle a\rangle$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Divergence en π-calcul

- $D_1 = !a.\overline{a} \mid \overline{a} \longrightarrow D_1$
- $D_2 = !a.\overline{b} \mid !b.\overline{a} \mid \overline{a} \longrightarrow\longrightarrow D_2$
- $D_3 = c(x).!a.\overline{x} \mid \overline{a} \mid \overline{c}\langle b\rangle \mid \overline{c}\langle a\rangle \longrightarrow D_1 \mid \overline{c}\langle b\rangle$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Systèmes de type basés sur le poids

Niveaux et Poids

- [DengSangiorgi06]
- Les Types donnent un *niveau* à chaque nom:

$$c(x).\overline{x}\langle 8\rangle \rightsquigarrow x : \sharp^2\,\mathtt{nat}, c : \sharp^1\,(\sharp^2\,\mathtt{nat})$$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Systèmes de type basés sur le poids

Niveaux et Poids

- [DengSangiorgi06]
- Les Types donnent un *niveau* à chaque nom:

$$c(x).\overline{x}\langle 8\rangle \rightsquigarrow x : \sharp^2\,\mathtt{nat}, c : \sharp^1\,(\sharp^2\,\mathtt{nat})$$

Contrôler les boucles

- Poids d'un processus, $\Gamma \vdash P : n$, niveau maximum d'un output dans l'état courant (partie non-répliquée).

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Systèmes de type basés sur le poids

Niveaux et Poids

- [DengSangiorgi06]
- Les Types donnent un *niveau* à chaque nom:

$$c(x).\overline{x}\langle 8\rangle \rightsquigarrow x : \sharp^2\,\mathtt{nat}, c : \sharp^1\,(\sharp^2\,\mathtt{nat})$$

Contrôler les boucles

- Poids d'un processus, $\Gamma \vdash P : n$, niveau maximum d'un output dans l'état courant (partie non-répliquée).
- Controle des réplications:
  - $!a^k(x).P$ avec $\vdash P : n$ requiert $k > n$.
- Correction: mesure décroissante.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemples

Exemple de réduction

- $T_1 = !a\ .(\overline{b}\ \mid \overline{b}\ \mid \overline{c}\ ) \mid !b\ .(\overline{c}\ \mid \overline{c}\ )$
- $T_1 \mid \overline{a} \mid \overline{b} \rightarrow T_1 \mid \overline{a} \mid \overline{c} \mid \overline{c} \rightarrow T_1 \mid \overline{b} \mid \overline{b} \mid \overline{c} \mid \overline{c} \mid \overline{c} \rightarrow\rightarrow\not\rightarrow$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemples

Exemple de réduction

- $T_1 = !a^3.(\overline{b}^2 \mid \overline{b}^2 \mid \overline{c}^1) \mid !b^2.(\overline{c}^1 \mid \overline{c}^1)$
- $T_1 \mid \overline{a} \mid \overline{b} \rightarrow_2 T_1 \mid \overline{a} \mid \overline{c} \mid \overline{c} \rightarrow_3 T_1 \mid \overline{b} \mid \overline{b} \mid \overline{c} \mid \overline{c} \mid \overline{c} \rightarrow_2 \rightarrow_2 \not\rightarrow$
- $\{3,2\} \rightarrow_2 \{3,1,1\} \rightarrow_3 \{2,2,1,1,1\} \rightarrow_2 \{2,1,1,1,1,1\} \rightarrow_2 \{1,1,1,1,1,1,1\}$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemples

Exemple de réduction

- $T_1 = !a^3.(\overline{b}^2 \mid \overline{b}^2 \mid \overline{c}^1) \mid !b^2.(\overline{c}^1 \mid \overline{c}^1)$
- $T_1 \mid \overline{a} \mid \overline{b} \rightarrow_2 T_1 \mid \overline{a} \mid \overline{c} \mid \overline{c} \rightarrow_3 T_1 \mid \overline{b} \mid \overline{b} \mid \overline{c} \mid \overline{c} \mid \overline{c} \rightarrow_2 \rightarrow_2 \not\rightarrow$
- $\{3,2\} \rightarrow_2 \{3,1,1\} \rightarrow_3 \{2,2,1,1,1\} \rightarrow_2 \{2,1,1,1,1,1\} \rightarrow_2 \{1,1,1,1,1,1,1\}$

Les Exemples divergents sont rejetés

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$
- $D_2 = !a^n.\overline{b}^k \mid !b^k.\overline{a}^n \mid \overline{a}^n$.
- $D_3 = c^t(x).!a^n.\overline{x}^k \mid \overline{a}^n \mid \overline{c}^t\langle a \rangle \mid \overline{c}^t\langle b \rangle$.

.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemples

Exemple de réduction

- $T_1 = !a^3.(\overline{b}^2 \mid \overline{b}^2 \mid \overline{c}^1) \mid !b^2.(\overline{c}^1 \mid \overline{c}^1)$
- $T_1 \mid \overline{a} \mid \overline{b} \rightarrow_2 T_1 \mid \overline{a} \mid \overline{c} \mid \overline{c} \rightarrow_3 T_1 \mid \overline{b} \mid \overline{b} \mid \overline{c} \mid \overline{c} \mid \overline{c} \rightarrow_2 \rightarrow_2 \not\rightarrow$
- $\{3,2\} \rightarrow_2 \{3,1,1\} \rightarrow_3 \{2,2,1,1,1\} \rightarrow_2 \{2,1,1,1,1,1\} \rightarrow_2 \{1,1,1,1,1,1,1\}$

Les Exemples divergents sont rejetés

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ pas typable: $n > n$
- $D_2 = !a^n.\overline{b}^k \mid !b^k.\overline{a}^n \mid \overline{a}^n$.
- $D_3 = c^t(x).!a^n.\overline{x}^k \mid \overline{a}^n \mid \overline{c}^t\langle a \rangle \mid \overline{c}^t\langle b \rangle$. .

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemples

Exemple de réduction

- $T_1 = !a^3.(\overline{b}^2 \mid \overline{b}^2 \mid \overline{c}^1) \mid !b^2.(\overline{c}^1 \mid \overline{c}^1)$
- $T_1 \mid \overline{a} \mid \overline{b} \rightarrow_2 T_1 \mid \overline{a} \mid \overline{c} \mid \overline{c} \rightarrow_3 T_1 \mid \overline{b} \mid \overline{b} \mid \overline{c} \mid \overline{c} \mid \overline{c} \rightarrow_2 \rightarrow_2 \not\rightarrow$
- $\{3,2\} \rightarrow_2 \{3,1,1\} \rightarrow_3 \{2,2,1,1,1\} \rightarrow_2 \{2,1,1,1,1,1\} \rightarrow_2 \{1,1,1,1,1,1,1\}$

Les Exemples divergents sont rejetés

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ pas typable: $n > n$
- $D_2 = !a^n.\overline{b}^k \mid !b^k.\overline{a}^n \mid \overline{a}^n$ pas typable: $n > k > n$.
- $D_3 = c^t(x).!a^n.\overline{x}^k \mid \overline{a}^n \mid \overline{c}^t\langle a \rangle \mid \overline{c}^t\langle b \rangle$.

.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemples

Exemple de réduction

- $T_1 = !a^3.(\overline{b}^2 \mid \overline{b}^2 \mid \overline{c}^1) \mid !b^2.(\overline{c}^1 \mid \overline{c}^1)$
- $T_1 \mid \overline{a} \mid \overline{b} \rightarrow_2 T_1 \mid \overline{a} \mid \overline{c} \mid \overline{c} \rightarrow_3 T_1 \mid \overline{b} \mid \overline{b} \mid \overline{c} \mid \overline{c} \mid \overline{c} \rightarrow_2 \rightarrow_2 \not\rightarrow$
- $\{3,2\} \rightarrow_2 \{3,1,1\} \rightarrow_3 \{2,2,1,1,1\} \rightarrow_2 \{2,1,1,1,1,1\} \rightarrow_2 \{1,1,1,1,1,1,1\}$

Les Exemples divergents sont rejetés

- $D_1 = !a^n.\overline{a}^n \mid \overline{a}^n$ pas typable: $n > n$
- $D_2 = !a^n.\overline{b}^k \mid !b^k.\overline{a}^n \mid \overline{a}^n$ pas typable: $n > k > n$.
- $D_3 = c^t(x).!a^n.\overline{x}^k \mid \overline{a}^n \mid \overline{c}^t\langle a \rangle \mid \overline{c}^t\langle b \rangle$.
  $c : \sharp^t\ (\sharp^n\ T)$ ainsi $n = k$ et $n > k$.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Système de types

$$(\textbf{Nil})\frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad (\textbf{Par})\frac{\Gamma \vdash P_1 : n_1 \qquad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)}$$

$$(\textbf{Res})\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \sharp^k\ T}{\Gamma \vdash (\nu a)\ P : n} \qquad (\textbf{Out})\frac{\Gamma \vdash P : n \qquad a : \sharp^k\ T \qquad v : T}{\Gamma \vdash \overline{a}\langle v \rangle.P : \max(n, k)}$$

$$(\textbf{In})\frac{\Gamma \vdash P : n \qquad a : \sharp^k\ T \qquad x : T}{\Gamma \vdash a(x).P : n}$$

$$(\textbf{Rep})\frac{\Gamma \vdash P : n \qquad a : \sharp^k\ T \qquad x : T \qquad k > n}{\Gamma \vdash !a(x).P : 0}$$

Système de types

$$(\textbf{Nil})\frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad (\textbf{Par})\frac{\Gamma \vdash P_1 : n_1 \qquad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)}$$

$$(\textbf{Res})\frac{\Gamma \vdash P : n \qquad \Gamma(a) = \sharp^k\ T}{\Gamma \vdash (\nu a)\ P : n} \qquad (\textbf{Out})\frac{\Gamma \vdash P : n \qquad a : \sharp^k\ T \qquad v : T}{\Gamma \vdash \overline{a}\langle v \rangle.P : \max(n, k)}$$

$$(\textbf{In})\frac{\Gamma \vdash P : n \qquad a : \sharp^k\ T \qquad x : T}{\Gamma \vdash a(x).P : n}$$

$$(\textbf{Rep})\frac{\Gamma \vdash P : n \qquad a : \sharp^k\ T \qquad x : T \qquad k > n}{\Gamma \vdash !a(x).P : 0}$$

Aller plus loin

- Comparer des ensembles de canaux lors des séquences d'inputs.
- $!a(x).b(y).c.c.(\overline{a}\langle x \rangle \mid \overline{x} \mid \overline{y}.\overline{c})$
  - Input reçus $>_{\text{mul}}$ Output émis
  - $\{a^4, b^3, c^1, c^1\} >_{\text{mul}} \{a^4, y^2, x^2, c^1\}$

Types linéaires

- Objectif: contrôler l'usage d'une ressource:
  - une communication exactement peut avoir lieu sur les canaux linéaires.
- Idées:
  - trois modes: linéaire $\sharp^0\ T$, affine $\sharp^1\ T$, répliqué $\sharp^\infty\ T$.
  - trois sortes de types: $\sharp^{\text{-}}\ T$, $\uparrow^{\text{-}}\ T$, $\downarrow^{\text{-}}\ T$
- séparation de contextes: $\Gamma = \Gamma_1 \oplus \Gamma_2$
  - si $c : \sharp^0\ T$ dans Γ,
    - soit $c : \sharp^0\ T$ dans Γ_1 (et pas dans Γ_2),
    - soit $c : \sharp^0\ T$ dans Γ_2 (et pas dans Γ_1),
    - soit $c :\downarrow^0\ T$ dans Γ_1 et $c :\uparrow^0\ T$ dans Γ_2,
    - soit $c :\uparrow^0\ T$ dans Γ_1 et $c :\downarrow^0\ T$ dans Γ_2,
  - si $c :\downarrow^0\ T$ dans Γ,
    - soit $c :\downarrow^0\ T$ dans Γ_1 (et pas dans Γ_2),
    - soit $c :\downarrow^0\ T$ dans Γ_2 (et pas dans Γ_1),
  - pareil pour $c :\uparrow^0\ T$
  - sinon, si $c : \sharp^{1,\infty}\ T$ dans Γ, $c : \sharp^{1,\infty}\ T$ dans Γ_1 et dans Γ_2

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Règle des types linéaires

$$(\textbf{Nil})\frac{\forall a \in \Gamma, \Gamma(a) = \sharp^{1,\infty}\ T}{\Gamma \vdash 0} \qquad (\textbf{Par})\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2 \qquad \Gamma = \Gamma_1 \oplus \Gamma_2}{\Gamma \vdash P_1 \mid P_2}$$

$$(\textbf{Out})\frac{\Gamma \vdash P \qquad \Gamma(a) = \sharp^{1,\infty}\ T \qquad \Gamma(v) = T \qquad T \text{ non-linéaire}}{\Gamma \vdash \overline{a}\langle v\rangle.P}$$

$$(\textbf{OutL})\frac{\Gamma \vdash P \qquad \Gamma(v) = T \qquad T \text{ non-linéaire}}{\Gamma, a :\uparrow^0 T \vdash \overline{a}\langle v\rangle.P}$$

$$(\textbf{LOut})\frac{\Gamma \vdash P \qquad \Gamma(a) = \sharp^{1,\infty}\ T \qquad T \text{ linéaire}}{\Gamma, v : T \vdash \overline{a}\langle v\rangle.P}$$

$$(\textbf{LOutL})\frac{\Gamma \vdash P \qquad T \text{ linéaire}}{\Gamma, a :\uparrow^0 T, v : T \vdash \overline{a}\langle v\rangle.P} \qquad (\textbf{In})\frac{\Gamma, x : T \vdash P \qquad \Gamma(a) = \sharp^{1,\infty}\ T}{\Gamma \vdash a(x).P}$$

$$(\textbf{InL})\frac{\Gamma, x : T \vdash P \qquad \Gamma(a) = \sharp^{1,\infty}\ T}{\Gamma, a :\downarrow^0 T \vdash a(x).P} \qquad (\textbf{Rep})\frac{\Gamma, x : T \vdash P \qquad \Gamma(a) = \sharp^{\infty}\ T}{\Gamma \vdash !a(x).P}$$

$$(\textbf{Res})\frac{\Gamma, a : T \vdash P}{\Gamma \vdash (\nu a)\ P}$$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemples

- $!a(x).(\overline{x} \mid x) \mid \overline{a}\langle b\rangle \mid \overline{a}\langle c\rangle$ typable avec $a : \sharp^{\infty}\,(\sharp^{0}\,())$
- $!a(x).(\overline{x}) \mid \overline{a}\langle b\rangle \mid \overline{a}\langle c\rangle$ pas typable.
- $!a(x).(\overline{x} \mid \overline{x} \mid x) \mid \overline{a}\langle b\rangle \mid \overline{a}\langle c\rangle$ pas typable.
- $!a(x).(\overline{x} \mid x) \mid \overline{a}\langle b\rangle \mid \overline{a}\langle b\rangle$ pas typable.
- $!a(x).(\overline{d}\langle x\rangle \mid x) \mid \overline{a}\langle b\rangle \mid \overline{a}\langle b\rangle \mid !d(y).\overline{y}$ typable avec $a : \sharp^{\infty}\,(\sharp^{0}\,())$ et $d : \sharp^{\infty}\,(\uparrow^{0}\,())$.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Vérification de Protocoles acentralisés

San Diego

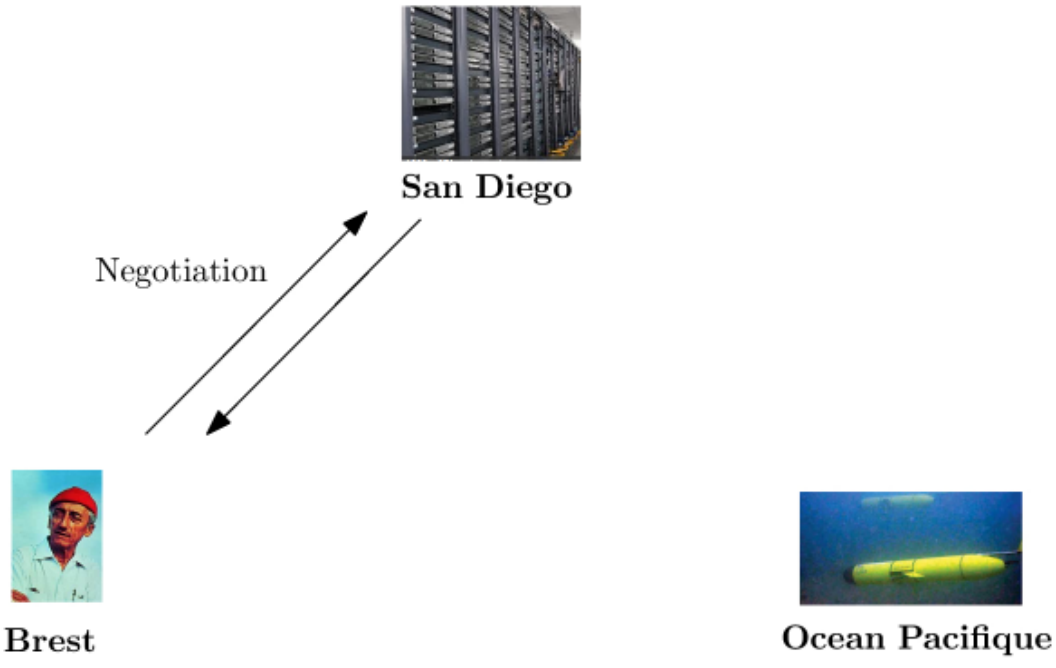
Brest

Ocean Pacifique

- Trois programmes indépendants (client, agent, instrument):
  - écrits dans des langages différents ,
  - avec librairies et compilateurs locaux,
  - interagissent par messages.
- Pas de contrôle global.
- Objectif: garantir le succès des interactions.
  - Méthode formelle: types de sessions.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Vérification de Protocoles acentralisés



- Trois programmes indépendants (client, agent, instrument):
  - écrits dans des langages différents ,
  - avec librairies et compilateurs locaux,
  - interagissent par messages.
- Pas de contrôle global.
- Objectif: garantir le succès des interactions.
  - Méthode formelle: types de sessions.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Vérification de Protocoles acentralisés

San Diego

Initialisation

Brest

Ocean Pacifique

- Trois programmes indépendants (client, agent, instrument):
  - écrits dans des langages différents ,
  - avec librairies et compilateurs locaux,
  - interagissent par messages.
- Pas de contrôle global.
- Objectif: garantir le succès des interactions.
  - Méthode formelle: types de sessions.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Vérification de Protocoles acentralisés

San Diego

Brest

Streaming

Ocean Pacifique

- Trois programmes indépendants (client, agent, instrument):
  - écrits dans des langages différents ,
  - avec librairies et compilateurs locaux,
  - interagissent par messages.
- Pas de contrôle global.
- Objectif: garantir le succès des interactions.
  - Méthode formelle: types de sessions.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Types de Sessions Binaires

- *Languages Primitives and Type Discipline for Structured Communication-Based Programming*, Honda, Kubo, Vasconcelos, ESOP 1998
  - Domaine: algèbres de processus (π-calcul): les agents communiquent avec des messages sur des canaux.
  - Motivation: construire des types pour guider les interactions entre deux agents sur un même canal.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Types de Sessions Binaires

- *Languages Primitives and Type Discipline for Structured Communication-Based Programming*, Honda, Kubo, Vasconcelos, ESOP 1998
  - Domaine: algèbres de processus (π-calcul): les agents communiquent avec des messages sur des canaux.
  - Motivation: construire des types pour guider les interactions entre deux agents sur un même canal.
- Principes:
  - Décrire formellement les interactions entre deux participants (une *session*) sur un unique canal s.
    - Briques de bases: communications (direction, étiquette, type du message), choix, récursion, fin de session.
  - Séparation du type en deux extrémités symétriques (semblables à des processus CCS).
  - Validation, (système de types) de chaque participant par rapport à son type respectif.
- Originalité dans les types pour π: la séquence:
  - Types simples pour le π-calcul: $a : \sharp^{i}(\mathtt{nat}, \sharp^{o}\,\mathtt{B})$.
  - Types de session: $s : ?(\mathtt{nat}); !(\mathtt{B})$.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Types de sessions binaires - Exemple

- Type global:

$$\begin{array}{lll} \texttt{Seller} \to \texttt{Buyer} & (\texttt{price}); & \\ \texttt{Buyer} \to \texttt{Seller} & -(\texttt{ko}); & \texttt{end} \\ & -(\texttt{ok}); & \{\texttt{Seller} \to \texttt{Buyer}\ (\texttt{item}); \\ & & \texttt{end}\} \end{array}$$

- Types locaux (extrémités):
  - $\texttt{Buyer} : ?(\texttt{price}); !\{\texttt{ko}; \texttt{end}\ ,\ \texttt{ok}; ?\texttt{item}; \texttt{end}\}$
  - $\texttt{Seller} : !(\texttt{price}); ?\{\texttt{ko}; \texttt{end}\ ,\ \texttt{ok}; !\texttt{item}; \texttt{end}\}$
- Processus candidats:
  - $s_{\texttt{price}}(p).(\overline{s}_{\texttt{ok}}.s_{\texttt{item}}(i) + \overline{s}_{\texttt{ko}})$:

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Types de sessions binaires - Exemple

- Type global:

$$\begin{array}{lll} \texttt{Seller} \to \texttt{Buyer} & (\texttt{price}); & \\ \texttt{Buyer} \to \texttt{Seller} & -(\texttt{ko}); & \texttt{end} \\ & -(\texttt{ok}); & \{\texttt{Seller} \to \texttt{Buyer}\ (\texttt{item}); \\ & & \texttt{end}\} \end{array}$$

- Types locaux (extrémités):
  - $\texttt{Buyer} : ?(\texttt{price}); !\{\texttt{ko}; \texttt{end}\ ,\ \texttt{ok}; ?\texttt{item}; \texttt{end}\}$
  - $\texttt{Seller} : !(\texttt{price}); ?\{\texttt{ko}; \texttt{end}\ ,\ \texttt{ok}; !\texttt{item}; \texttt{end}\}$
- Processus candidats:
  - $s_{\texttt{price}}(p).(\overline{s}_{\texttt{ok}}.s_{\texttt{item}}(i) + \overline{s}_{\texttt{ko}})$: bon $\texttt{Buyer}$.
  - $s_{\texttt{price}}(p).\overline{s}_{\texttt{ko}}$:

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Types de sessions binaires - Exemple

- Type global:

$$\begin{array}{lll} \texttt{Seller} \to \texttt{Buyer} & (\texttt{price}); & \\ \texttt{Buyer} \to \texttt{Seller} & -(\texttt{ko}); & \texttt{end} \\ & -(\texttt{ok}); & \{\texttt{Seller} \to \texttt{Buyer}\ (\texttt{item}); \\ & & \texttt{end}\} \end{array}$$

- Types locaux (extrémités):
  - $\texttt{Buyer} : ?(\texttt{price}); !\{\texttt{ko}; \texttt{end}\ ,\ \texttt{ok}; ?\texttt{item}; \texttt{end}\}$
  - $\texttt{Seller} : !(\texttt{price}); ?\{\texttt{ko}; \texttt{end}\ ,\ \texttt{ok}; !\texttt{item}; \texttt{end}\}$
- Processus candidats:
  - $s_{\texttt{price}}(p).(\overline{s}_{\texttt{ok}}.s_{\texttt{item}}(i) + \overline{s}_{\texttt{ko}})$: bon Buyer.
  - $s_{\texttt{price}}(p).\overline{s}_{\texttt{ko}}$: bon Buyer.
  - $\overline{s}_{\texttt{price}}\langle 100\ \texttt{Fr}\rangle.s_{\texttt{ko}}$:

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Types de sessions binaires - Exemple

- Type global:

$$\begin{array}{ll} \texttt{Seller} \to \texttt{Buyer} & (\texttt{price}); \\ \texttt{Buyer} \to \texttt{Seller} & -(\texttt{ko}); \quad \texttt{end} \\ & -(\texttt{ok}); \quad \{\texttt{Seller} \to \texttt{Buyer}\ (\texttt{item}); \\ & \qquad\qquad\ \ \texttt{end}\} \end{array}$$

- Types locaux (extrémités):
  - $\texttt{Buyer} : ?(\texttt{price}); !\{\texttt{ko}; \texttt{end}\ ,\ \texttt{ok}; ?\texttt{item}; \texttt{end}\}$
  - $\texttt{Seller} : !(\texttt{price}); ?\{\texttt{ko}; \texttt{end}\ ,\ \texttt{ok}; !\texttt{item}; \texttt{end}\}$
- Processus candidats:
  - $s_{\texttt{price}}(p).(\overline{s}_{\texttt{ok}}.s_{\texttt{item}}(i) + \overline{s}_{\texttt{ko}})$: bon $\texttt{Buyer}$.
  - $s_{\texttt{price}}(p).\overline{s}_{\texttt{ko}}$: bon $\texttt{Buyer}$.
  - $\overline{s}_{\texttt{price}}\langle 100\ \texttt{Fr}\rangle.s_{\texttt{ko}}$: mauvais $\texttt{Seller}$.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Vers les MPST

- Les types extrémités sont parfaitement symétriques.
  - Ils s'assurent que les deux partis s'entendent sur les actions à effectuer.
- Les types de sessions binaires peuvent être mis en relation avec les logiques linéaires/intuitionnistes:
  - *Session Types as Intuitionistic Linear Propositions*, Caires, Pfenning, CONCUR 2010
  - Beaucoup de développement récents dans les dernières années: "Curry-Howard for sessions".
- Challenge: Les protocoles dans les réseaux impliquent souvent plus de deux participants:
  - la symétrie est cassée,
  - on introduit de l'asynchronie:
    - $\mathtt{A} \to \mathtt{B}(\mathrm{m}_1); \mathtt{A} \to \mathtt{C}(\mathrm{m}_2); \mathtt{C} \to \mathtt{B}(\mathrm{m}_3)$,
    - B peut recevoir m_3 avant de recevoir m_1.
  - *Multiparty Asynchronous Session Types*, Carbone, Honda, Yoshida, POPL 2008

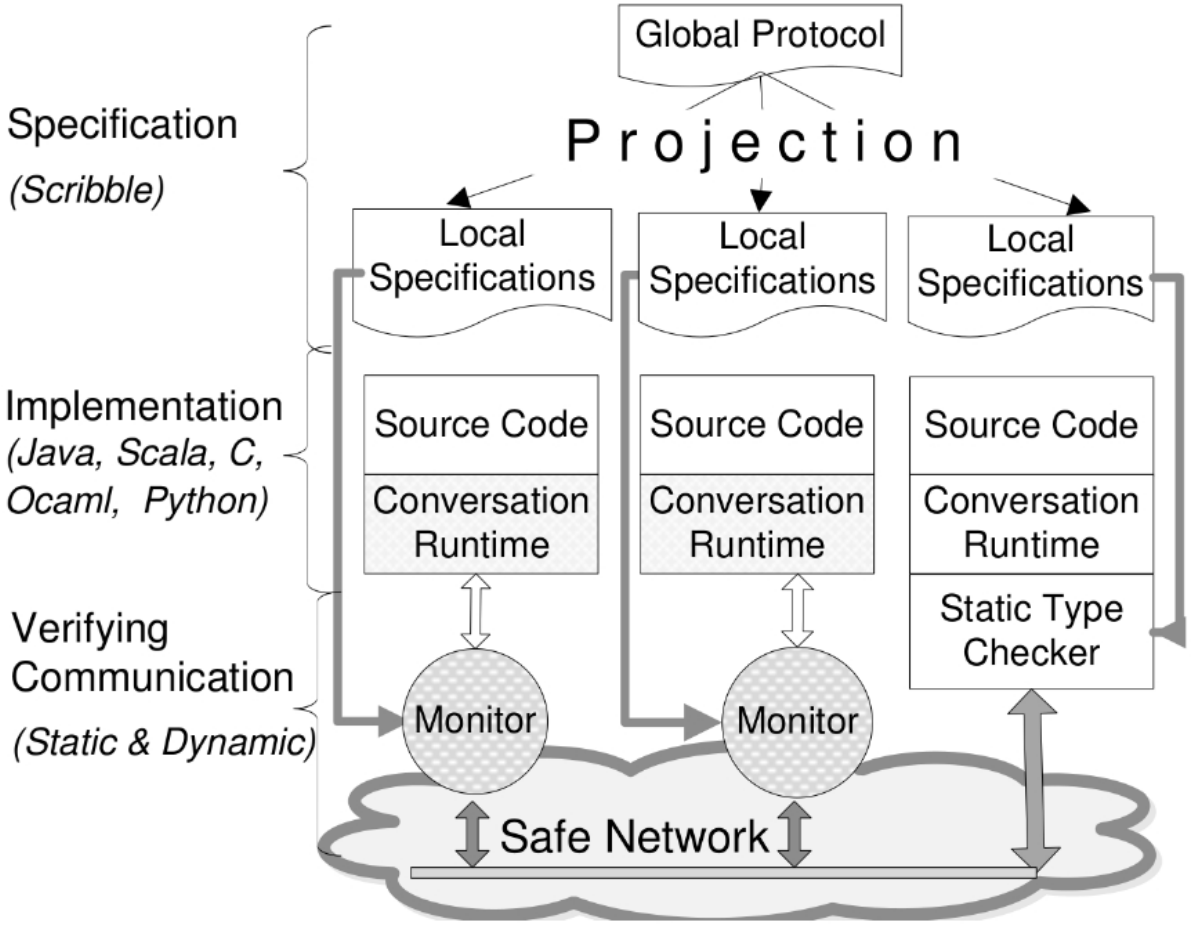
SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

MPST comme Méthode de vérification

- Vérification pour des réseaux de services/applications:
  - réseaux acentralisés
    - communication par passage de message,
    - pas de contrôle global.
  - spécification: chorégraphies globales d'interactions entre plusieurs participants
    - des rôles interagissent dans une session.
    - les types globaux sont projetés en types locaux.
  - Thm: les agents suivent localement leurs types locaux
    $\Rightarrow$ le réseau suit globalement la spécification.
- Vérifier les types locaux aux extrêmités:
  - validation: analyser statiquement le programme (*typechecker*).
  - *monitoring*: analyser à la volée les messages entrants et sortants de l'application.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

MPST comme Méthode de vérification (II)

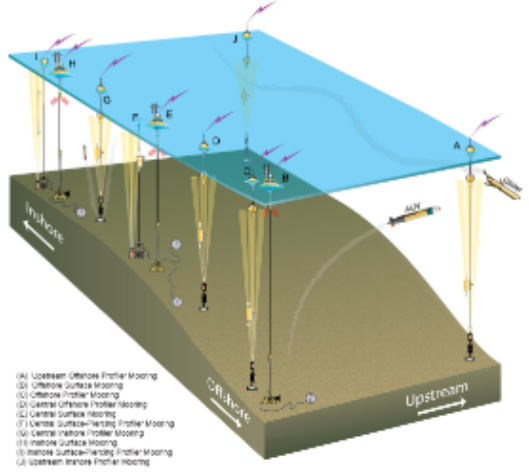


(extrait de *Monitoring Networks through Multiparty Session Types*)

Collaborations

- Réseaux de services, web.
  - difficulté d'obtenir des applications concrètes.
- Intéressants pour les banques (BoJ, UBS, JP Morgan)
  - propriétaires de larges infrastructures distribuées,
  - accent mis sur la correction, la sécurité.
- Principale collaboration: *Ocean Observatory Initiative*
  - Projet international d'océanographie.
  - Communication par passage de messages dans de grandes infrastructures.

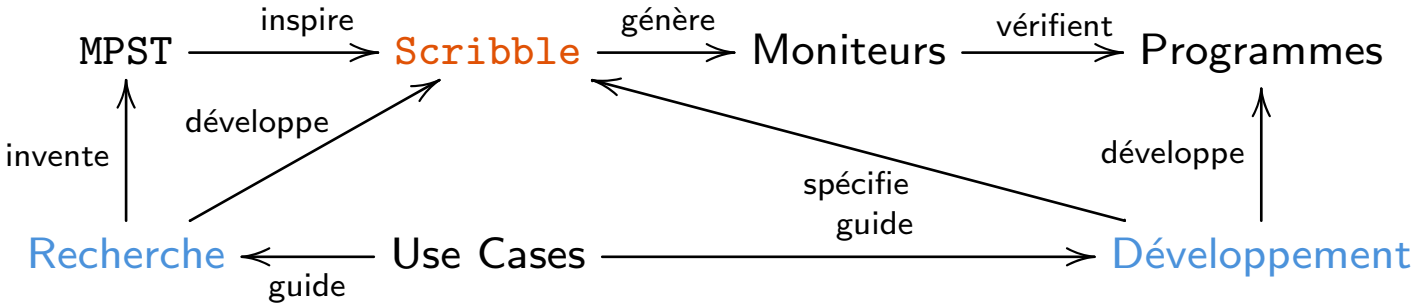
OCEAN OBSERVATORIES INITIATIVE



SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Collaboration avec OOI

- Développement d'un langage de protocoles (*Scribble*).
- Cycle de vie d'une *feature* (e.g. exceptions, modularité):
  - Echange avec les développeurs et les océanographes, validation par les chefs de projet.
  - Développement d'une théorie formelle (π-calculs) avec preuves, publication.
  - Intégration au langage *Scribble*: sémantique (mot-clefs), moniteurs, toolsuite.
  - Utilisation par les développeurs (spécification et création de moniteurs).



MPST: Théorie, types globaux

$$
\begin{array}{lcll}
G & ::= & \mathtt{r}_1 \to \mathtt{r}_2 : \Sigma_{i \in I} \{ l_i(x_i : S_i); G_i \} \mid \mathtt{end} & \text{(com,fin)} \\
 & \mid & G_1 \oplus^{\mathtt{r}} G_2 \mid G_1 \parallel G_2 & \text{(choix,par)} \\
 & \mid & \mu \mathtt{t}.G \mid \mathtt{t} & \text{(rec,rec-var)}
\end{array}
$$

- $\mathtt{r}$: rôles (participant d'une session).
- li: etiquettes des communications.
- S_i: type du message.
- $\oplus^{\mathtt{r}}$: choix d'un participant $\mathtt{r}$.
- Récursion par variables $\mathtt{t}$.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

MPST: Types Globaux - Exemples

$$
\begin{array}{l}
\mathtt{A} \to \mathtt{B} : \mathtt{ask}(x : \mathtt{title});\\
\mathtt{A} \to \mathtt{C} : \mathtt{ask}(x : \mathtt{title});\\
\mathtt{B} \to \mathtt{A}; \mathtt{price}(y_1 : \mathtt{nat});\\
\mathtt{C} \to \mathtt{A}; \mathtt{price}(y_2 : \mathtt{nat});\\
(\mathtt{A} \to \mathtt{B} : \mathtt{ok};\\
\ \mathtt{A} \to \mathtt{C} : \mathtt{ko};\\
\ \mathtt{B} \to \mathtt{A} : \mathtt{movie}(m_1 : .\mathtt{avi}); \mathtt{end}\\
) \oplus^{\mathtt{A}} (\\
\ \mathtt{A} \to \mathtt{C} : \mathtt{ok};\\
\ \mathtt{A} \to \mathtt{B} : \mathtt{ko};\\
\ \mathtt{C} \to \mathtt{A} : \mathtt{movie}(m_2 : .\mathtt{avi}); \mathtt{end})
\end{array}
$$

- Asynchronie: $\mathtt{A}$ peut recevoir y_2 de $\mathtt{C}$ avant y_1 de $\mathtt{B}$.
- Conditions de bonne formation:

$$
\begin{array}{l}
(\mathtt{A} \to \mathtt{B} : \mathtt{hello}; \mathtt{D} \to \mathtt{A} : \mathtt{black})\\
\oplus^{\mathtt{A}} (\mathtt{A} \to \mathtt{C} : \mathtt{hello}; \mathtt{D} \to \mathtt{A} : \mathtt{white})
\end{array}
$$

est mal-formé.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Types Locaux

Obtenus par projection automatique des types globaux:

$$\begin{array}{rcl} T & ::= & \mathtt{r}?_{i\in I}\{l_i(x_i : S_i); T_i\} \mid \mathtt{r}!_{i\in I}\{l_i(x_i : S_i); T_i\} \\ & \mid & T \parallel T \mid T \oplus T \mid \mu \mathtt{t}.T \mid \mathtt{t} \mid \mathtt{end} \end{array}$$

- $\mathtt{r}?_{i\in I}\{l_i(x_i : S_i); T_i\}$: reçoit de $\mathtt{r}$, projection d'une communication sur le receveur.
- $\mathtt{r}!_{i\in I}\{l_i(x_i : S_i); T_i\}$: envoie à $\mathtt{r}$, projection d'une communication sur l'émetteur.
- Exemple: projection sur A:

$T_{\mathtt{A}}$: $\mathtt{B}!\mathtt{ask}(x : \mathtt{title}); \mathtt{C}!\mathtt{ask}(x : \mathtt{title}); \mathtt{B}?\mathtt{price}(y_1 : \mathtt{nat}); \mathtt{C}?\mathtt{price}(y_2 : \mathtt{nat});$
$(\mathtt{B}!\mathtt{ok}; \mathtt{C}!\mathtt{ko}; \mathtt{B}?\mathtt{movie}(m_1 : .\mathtt{avi}); \mathtt{end})$
$\oplus(\mathtt{C}!\mathtt{ok}; \mathtt{B}!\mathtt{ko}; \mathtt{C}?\mathtt{movie}(m_2 : .\mathtt{avi}); \mathtt{end})$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Session-calcul

$$
\begin{array}{rcl}
P & ::= & \mathbf{0} \mid P|P \mid a(x).P \mid \overline{a}\langle s\rangle.P \mid P+P \\
& \mid & k?[\mathtt{r},\mathtt{r}]_{i\in I}\{l_i(x_i).P_i\} \mid k![\mathtt{r},\mathtt{r}]l\langle v\rangle.P \mid (\nu u)\ P \\
& \mid & \mu X(x).P\langle v\rangle \mid X\langle v\rangle
\end{array}
$$

- a: *canaux partagés*, utilisés pour les invitations,
- k, s: canaux de sessions.

Réductions (+ congruence):

$$
\frac{}{s![\mathtt{r}_1,\mathtt{r}_2]l_j\langle\widetilde{v}\rangle.P \mid s?[\mathtt{r}_1,\mathtt{r}_2]_{i\in I}\{l_i(\widetilde{x_i}).P_i\} \longrightarrow P \mid P_j\{\widetilde{v}/\widetilde{x_j}\}}
$$

$$
\frac{}{\overline{a}\langle\widetilde{v}\rangle.P \mid a(\widetilde{y}).Q \longrightarrow P \mid Q\{\widetilde{v}/\widetilde{y}\}}
$$

$$
\frac{P \longrightarrow P'}{(\nu\widetilde{a})\ (P \mid R) \longrightarrow (\nu\widetilde{a})\ (P' \mid R)}
$$

$$
\frac{P_i \longrightarrow P_i'}{(P_1+P_2) \longrightarrow P_i'}
$$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Système de Types

- Jugements $\Gamma \vdash P \triangleright \Delta$
  - Γ: environnement global, connaissance initiale, (canaux partagés, ...)
  - Δ: environnement local (types de sessions)
    - $s[\mathtt{r}]^{\bullet}$: type en attente (va être envoyé à quelqu'un)

$$\frac{(\Gamma \vdash P_i \triangleright \Delta, s[\mathtt{r}'] : T_i \qquad \vdash y_i : S_i)_{i \in I}}{\Gamma \vdash s?[\mathtt{r}, \mathtt{r}']_{i \in I}\{l_i(y_i).P_i\} \triangleright \Delta, s[\mathtt{r}'] : \mathtt{r}?_{i \in I}\{l_i(x_i : S_i).T_i\}}$$

- Typage de la réception $s?[\mathtt{r}, \mathtt{r}']_{i \in I}\{l_i(y_i).P_i\}$
  - choix entre plusieurs branches P_i déterminé par l'étiquette reçue l_i.
  - même structure entre le type et le processus.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Système de Types (Règles)

$$\frac{\Gamma \vdash P \triangleright \Delta, x[\mathrm{r}] : T \qquad \Gamma(a) = T[\mathrm{r}]}{\Gamma \vdash a(x).P \triangleright \Delta} \qquad \frac{\Gamma \vdash P \triangleright \Delta \qquad \Gamma(a) = T[\mathrm{r}]}{\Gamma \vdash \overline{a}\langle s \rangle.P \triangleright \Delta, s[\mathrm{r}]^{\bullet} : T}$$

$$\frac{(\Gamma \vdash P_i \triangleright \Delta, s[\mathrm{r}'] : T_i \qquad \vdash y_i : S_i)_{i \in I}}{\Gamma \vdash s?[\mathrm{r}, \mathrm{r}']_{i \in I}\{l_i(y_i).P_i\} \triangleright \Delta, s[\mathrm{r}'] : \mathrm{r}?_{i \in I}\{l_i(x_i : S_i).T_i\}}$$

$$\frac{\Gamma \vdash P \triangleright \Delta, s[\mathrm{r}] : T_j \qquad \vdash v : S_j}{\Gamma \vdash s![\mathrm{r}, \mathrm{r}']l_j\langle v \rangle.P \triangleright \Delta, s[\mathrm{r}] : \mathrm{r}'!_{i \in I}\{l_i(x_i : S_i).T_i\}}$$

$$\frac{}{\Gamma \vdash \mathbf{0} \triangleright \emptyset} \qquad \frac{\Gamma \vdash P_1 \triangleright \Delta_1 \qquad \Gamma \vdash P_2 \triangleright \Delta_2}{\Gamma \vdash P_1 \mid P_2 \triangleright \Delta_1 \otimes \Delta_2}$$

$$\frac{\Gamma \vdash P_1 \triangleright \Delta, s[\mathrm{r}] : T_1 \qquad \Gamma \vdash P_2 \triangleright \Delta, s[\mathrm{r}] : T_2}{\Gamma \vdash P_1 + P_2 \triangleright \Delta, s[\mathrm{r}] : T_1 \oplus T_2}$$

$$\frac{\Gamma \vdash P \triangleright \Delta, s[\mathrm{r}] : T_i \qquad i \in \{1, 2\}}{\Gamma \vdash P \triangleright \Delta, s[\mathrm{r}] : T_1 \oplus T_2} \qquad \frac{\Gamma, a : T[\mathrm{r}] \vdash P \triangleright \Delta}{\Gamma \vdash (\nu a)\ P \triangleright \Delta}$$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Scribble

```
include bbs_aux;

global protocol BuyerBrokerSupplier2(role Buyer, role Broker, role Supplier) {
    rec START {
        do ForwardQuery(Buyer, Broker, Supplier); // Perform as inline protocol
        do ForwardPrice(Supplier, Broker, Buyer);
        price(int) from Broker to Buyer;
        choice at Buyer {
            do ForwardRedo(Buyer, Broker, Supplier);
            continue START;
        } or {
            accept() from Buyer to Broker;
            confirm() from Broker to Supplier;
            do ForwardDate(Supplier, Broker, Buyer);
        } or {
            reject() from Buyer to Broker;
            cancel() from Broker to Supplier;
        }
    }
}
```

- Traduction directe des types de sessions formels.
- Utilisé par les développeurs pour spécifier des protocoles.
- Le code d'une application est vérifié par rapport à la spécification en Scribbe:
  - Outil de projection pour construire les types locaux,
  - Outil de création de moniteurs, FSMs qui tournent en même temps que les applications,
  - ou Typechecker qui valide des morceaux de code (Session C, OCaml).

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Expressivité des MPST

Etat de l'art.

- Expressivité interne (description de ce qui ce passe au sein une session):
  - structures de contrôle plus précises (automates): *Multiparty Session Types Meet Communicating Automata*, Denielou, Yoshida, ESOP 2012
  - vérifier des propriétés de sécurité: *Information Flow Safety in Multiparty Sessions*, Capecchi, Castellani, Dezani, EXPRESS 2011
  - prendre en compte les comportements exceptionnels: *Practical Interruptible Conversations - Distributed Dynamic Verification with Session Types and Python*, Hu, Neykova, Yoshida, D., RV 2013
  - assurer l'absence d'interbloquages: *Deadlock-freedom-by-design: multiparty asynchronous global programming*, Carbone, Montesi, POPL 2013

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Expressivité des MPST

- Expressivité externe (comment les sessions sont créées, rejointes):
  - appliquer un schéma d'invitations à plusieurs ensembles de participants: *Dynamic multirole session types*, Denielou, Yoshida, POPL 2011
  - gérer les autorisations de participations aux sessions: *Information Flow Safety in Multiparty Sessions*, Bono, Capecchi, Castellani, Dezani, TGC 2011
- Expressivité logique:
  - relation avec les logiques linéaires/intuitionnistes: *Propositions as Sessions*, Wadler, ICFP 2012
  - encodage dans le π-calcul typé: *Full abstraction in a subtyped π-calculus with linear types*, D., Honda, CONCUR 2012

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Réseaux de Types de Sessions

Monitoring Networks through Multiparty Session Types, Bocchi, Chen, D., Honda, Yoshida:, FMOODS/FORTE 2013

- Réseau: $[P_1]_{\alpha_1} \mid \ldots \mid [P_n]_{\alpha_n} \mid \langle Q; R \rangle$
- α: *principals*, participants, processus localisés.
  - Un principal participe à plusieurs sessions.
- $Q = s[\mathtt{r}_1, \mathtt{r}_2]\mathtt{price}\langle 100 \rangle, \ldots$: file globale pour les messages
  - messages adressés à des rôles.
- $R = s[\mathtt{r}_1] \mapsto \alpha_2, s[\mathtt{r}_2] \mapsto \alpha_1, k[\mathtt{r}_2] \mapsto \alpha_2, \ldots$: routeur, associe des rôles dans des sessions à des principaux.
- Equivalences: la *barbed congruence* introduit une notion d'interface: deux réseaux qui implémentent les mêmes services de manières différentes.
- Machines Abstraites pour les sessions.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Logique Multipartite Multisession

- Objectif: utiliser les types de sessions pour maintenir des propriétés logiques sur les messages et les participants.
- *A Theory of Design-by-Contract for Distributed Multiparty Interactions*, Bocchi, Honda, Tuosto, Yoshida, CONCUR 2010
- Sur les messages: $\mathtt{A} \to \mathtt{B} : \mathtt{price}(x)\{x < 100\}$
  - Conditions de bonne-formation complexes.
- *A Multiparty Multisession Logic*, Bocchi, D., Yoshida, TGC 2012
- Sur les messages et les participants:
  $\mathtt{A} \to \mathtt{B} : \mathtt{price}(x)\{x < \mathtt{A}.\mathrm{money}\}$
  - Cellule mémoire money du principal jouant le rôle de $\mathtt{A}$, maintenue à travers les sessions.
  - Invariants pour empêcher les courses sur les ressources..
- Encodage dans la logique de Hennessy-Milner.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exceptions dans les session types

Practical Interruptible Conversations - Distributed Dynamic Verification with Session Types and Python, Hu, Neykova, Yoshida, D., RV 2013

- Objectif: intégrer de manière sûre des comportements exceptionnels pour les sessions.
  - nécessaire pour les *use cases* de OOI: `abort`, `timeout`, ...
- spécification de blocs `interruptible` et des différents comportements exceptionnels qui peuvent être attrapés.
- Exceptions asynchrones:
  - $A \longrightarrow B; (A \longrightarrow D; \bullet\ A \longrightarrow D \parallel A \longrightarrow C; C \longrightarrow B; B \longrightarrow C)$
  - A interrompt à $\bullet$, comment interrompre l'autre branche du $\parallel$?
- Règles:
  - le message d'exception est broadcasté à tous les participants du bloc d'exception,
  - après émission ou réception d'un message d'exception, aucune réduction dans le bloc correspondant n'est possible.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Gouvernance en Scribble

- Stratégie à deux niveaux:
  - Un palier pour la gouvernance pure: syntax exprimant des engagements, contrats, promesses, options.
    - Par exemple: "en acceptation une requête du Client, le Serveur doit envoyer une réponse à l'Agent en moins de 10 secondes"
  - Un palier de contrôle des interactions: Scribble avec des assertions. Les communications sont annotées avec des assertions logiques.
- Scribble (avec assertions logiques) est suffisamment expressif.
  - Le premier palier est du sucre syntaxique.
  - Traduction automatisée d'un tier à l'autre.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Scribble avec Assertions

- Principes:
  - les moniteurs ont une mémoire (*ghost state*) qui persiste à travers les sessions (portefeuille, état, ...).
  - les communications sont annotées:
    - Assertions, propriété logiques sur le contenu des messages et l'état mémoire, qui sont vérifiées à l'envoi et à la réception.
    - Updates qui sont faites sur l'état local.
- $\texttt{money}(m : \mathrm{dollars})\ \texttt{from A to B}\{m = 100, \texttt{wallet@A} > 100\}\{\texttt{wallet@A}- = 100\}$
- le temps est représenté par des horloges (variables locales) (synchronisées ou non).

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Governance: Exemple

```
1  global protocol RPC(role Client, role Provider) {
2      time@Provider:sec,com@Provider:sec,$clock@Client,
3      req(t:sec) from Client to Provider;
4      choice at Provider {
5        accept(sig:Sig) from Provider to Client {sig=Sp(t)}{time:=$clock};
6        choice at Provider {
7            return() from Provider to Client {$clock<=time+t}{com:=success};
8             } or {
9            sorry() from Provider to Client {$clock>time+t}{com:=failure};
10           }
11     } or {
12     reject() from Provider to Client;
13 }
14
15.
```

- Variables locales déclarées en ligne 3.
- `$clock` vue comme une variable spéciale.
- `com` mis à jour $\Rightarrow$ journal.

Encoder les sessions dans le π-calcul

Full abstraction in a subtyped pi-calculus, D., Honda, CONCUR 2011

- Etude d'un π-calcule (sous-)typé concis:
  - $a\&\{l_1(x_1).P_1, \ldots, l_n(x_n).P_n\}$ and $\overline{a} \oplus \{l_1\langle v_1\rangle, \ldots, l_n\langle v_n\rangle\}$
- Encodage:
  - du λ-calcul simplement typé
  - calcul de sessions
- Encodage des sessions dans les types i/o simples.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Encodage - Principes

- Deux difficultés:
  - Emissions asynchrones: utilisation de noms administratifs pour garder les continuations:

$$[\![\overline{u}(v).P]\!] = (\nu v, c)\ (\overline{u}\langle v, c\rangle \mid c.[\![P]\!])$$

  - Transporter le futur d'un canal de session (sequentialité):
    - utiliser un nouveau nom de session s pour remplacer k à la prochaine étape de la session.
    - canaux administratifs utilisés pour transmettre le nouveau nom de la session:

$$[\![k!\langle 3\rangle.k?(x)]\!] = (\nu c)\ \overline{k}\langle 3, c\rangle \mid c(s).s(x)$$

  - Encodage des types:

$$[\![\uparrow(T).S]\!] = \uparrow([\![T]\!], \downarrow [\![S]\!])$$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Encodage - Termes

$$
\begin{array}{lcl}
[\![u(x).P]\!] & = & u(x,c).(\overline{c} \mid [\![P]\!]) \\
[\![\overline{u}(x).P]\!] & = & (\nu x, c)\ (\overline{u}\langle x, c\rangle \mid c.[\![P]\!]) \\
[\![k!l\langle e\rangle.P]\!] & = & (\nu c)\ (k \oplus^{\overline{1}} l\langle e, c\rangle \mid c(s).[\![P]\!]\{s/k\}) \\
[\![k?\{l_i(x_i).P_i\}_{i\in I}]\!] & = & (\boldsymbol{\nu} s)\ k\&^{\overline{1}}_{i\in I}\{l_i(x_i, c).([\![P_i]\!]\{s/k\} \mid \overline{c}\langle s\rangle)\}
\end{array}
$$

Ajoute un nom administratif et un nouveau nom de session à chaque communication.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Encodage - Exemple

Encodages pour un protocole binaire $\texttt{Buyer} - \texttt{Seller}$:

- Encodage des types locaux:

$$[\![\downarrow \texttt{offer}(\mathrm{Data}).\uparrow \left\{ \begin{array}{l} \texttt{yes}(\mathrm{Money}).\downarrow \texttt{movie}(.\mathrm{avi}).\texttt{end} \\ \texttt{no.end} \end{array} \right.]\!]$$

$$=\downarrow \texttt{offer}(\mathrm{Data}, \uparrow \oplus \left\{ \begin{array}{l} \texttt{yes}(\mathrm{Money}, \downarrow\downarrow \texttt{movie}(.\mathrm{avi}, \uparrow \bigstar) \\ \texttt{no}(\bigstar, \downarrow \bigstar) \end{array} \right.)$$

- Encodage des processus:

$$[\![s?\texttt{offer}(o).\overline{s}!\texttt{yes}\langle o.price\ \$\rangle.s?\texttt{movie}(m)]\!] =$$

$$(\nu s_2)\ s\&\texttt{offer}(o, c_1).[\overline{c_1}\langle s_2\rangle \mid (\nu c_2\ \overline{s_2} \oplus \texttt{yes}\langle o.price\ \$, c_2\rangle \mid c_2(s_3).(\nu s_4\ s_3\&\texttt{movie}(m, c_3).\overline{c_3}\langle s_4\rangle)]$$

Encodage - *Full abstraction* et sous-typage

Théorème

Soit P, Q deux $\pi^{\mathtt{session}}$ processus t.q. $\Gamma \vdash_{\pi} P$ et $\Gamma \vdash_{\pi} Q$, alors $P \simeq_{\mathtt{test}} Q$ ssi $[\![P]\!] \simeq_{\mathtt{test}} [\![Q]\!]$.

- $\simeq_{\mathtt{test}}$: *testing congruence* (may/must).
- Technique de preuve:
  - *Definability*: décoder les sessions dans un π-processus.
  - Adéquation:
    - P et $[\![P]\!]$ se comporte de la même façon.
    - Epurer les transitions administratives.

Sous-typage

Si $S_1 \leq S_2$, alors $[\![S_1]\!] \leq [\![S_2]\!]$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Sous-sessions, Modularité

- *Nesting Protocols in Session Types*, D, Honda, CONCUR 2012.
- Appeler explicitement un protocole dans un autre protocole.
- Issu des discussions sur les *use cases* au sein du projet OOI.
- les protocoles sont très modulaires:
  - ils sont paramètrés,
  - ils appellent d'autres protocoles.
- beaucoup de protocoles partagent la même forme.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Sous-protocoles

Negotiate

$\mathtt{r}_1 \to \mathtt{r}_2 : \mathrm{ask}(\mathit{terms}).$
$\mu \mathtt{t}.$
$\mathtt{r}_2 \to \mathtt{r}_1 : \mathrm{proposition}(\mathit{contract}_2).$
$\mathtt{r}_1 \to \mathtt{r}_2 : \{\mathrm{accept}.\mathtt{end}$
$\mathrm{counter}(\mathit{contract}_1).\mathtt{t}\}$

Resource_Usage

client $\to$ agent : request(*coord*). agent $\to$ instr : connect
instr $\to$ agent : available. agent $\to$ client : ack.
Negotiate between agent and client.
$\mu \mathtt{t}.$
client $\to$ instr : {abort(*coord*).end
command(*code*).
instr $\to$ client : result(*data*).t}

- *Negotiate* existe indépendamment, peut être modifié séparément.
- les valeurs dans *Negotiate* peuvent être instantiées de différentes manières.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Intérêts des protocoles modulaires

- donner une structure générique pour des protocoles similaires,
- exprimer une architecture de protocoles modulaire:
  - distinguer les sous-protocoles des protocoles principaux ,
  - abstraire l'implémentation login, tests de sécurité, . . .),
  - modifier les sous-protocoles de manière indépendante,
- décrire des protocoles de manière claire et concise,
  - appeler plusieurs copies du même protocole.
- sélectionner le bon nombre de participant,

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Principes

- un `let in` pour définir des protocoles.
- un `calls` pour appeler un protocole.
- deux invitations:
  - internes: participants de la supersession invités à la sous-session.
  - externes: nouveaux participants invités juste à la sous-session.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemple: Client-Middleware-Server

Protocole CMS avec deux participants: client et middle:

$$
\begin{array}{l}
\texttt{let } \mathit{Contact} = \lambda \text{agent}, \mathit{req} \mapsto \\
\qquad \texttt{new server}. \\
\qquad \text{agent} \to \text{server} : \text{request}(\mathit{req}). \\
\qquad \text{server} \to \text{agent} : \text{answer}(\mathit{ans}). \\
\qquad \texttt{end} \\
\texttt{in} \\
\text{client} \to \text{middle} : \text{request}(\mathit{req}_0). \\
\qquad (\text{middle} \to \text{client} : \text{answer}(\mathit{ans}_0).\texttt{end}) \\
\oplus^{\text{middle}} \\
\qquad (\text{middle } \texttt{calls } \mathit{Contact}(\text{middle}, \mathit{req}_0). \\
\qquad \text{middle} \to \text{client} : \text{answer}(\mathit{ans}_0).\texttt{end})
\end{array}
$$

- si le *middleware a la réponse*, le serveur n'est pas invité.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemple: Marché

$$
\begin{aligned}
&\texttt{let } \mathit{Buy} = \lambda \text{ agent} : \texttt{Role}, \text{seller} : \texttt{Role}, \text{item} : \texttt{Tradable} \mapsto \ldots \\
&\texttt{in let } \mathit{Sell} = \lambda \text{ agent} : \texttt{Role}, \text{buyer} : \texttt{Role}, \text{item} : \texttt{Tradable} \mapsto \ldots \\
&\texttt{in let } \mathit{Meet} = \lambda \text{ agent} : \texttt{Role}, \text{partner} : \texttt{Role}, \\
&\qquad\qquad\qquad\qquad \text{item} : \texttt{Tradable}, \mathit{Action} : (\texttt{Role} \to \texttt{Role} \to \texttt{Tradable} \to \diamond) \mapsto \ldots \\
&\text{agent } \texttt{calls } \mathit{Action}\langle \text{partner}, \text{item} \rangle \ldots \\
&\texttt{in } \ldots \\
&\text{alice } \texttt{calls } \mathit{Meet}\langle \text{bob}, \text{kettle}, \mathit{Buy} \rangle. \text{ carol } \texttt{calls } \mathit{Meet}\langle \text{bob}, \text{teacup}, \mathit{Sell} \rangle \ldots
\end{aligned}
$$

- Ordre supérieur (passage de protocoles).
- Utilisation de sortes pour la composition de protocoles:
  - *Buy*: $\texttt{Role} \to \texttt{Role} \to \texttt{Tradable} \to \diamond$
  - *Meet*: $\texttt{Role} \to \texttt{Role} \to \texttt{Tradable} \to (\texttt{Role} \to \texttt{Role} \to \texttt{Tradable} \to \diamond) \to \diamond$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Types locaux

Projetés des types globaux:

$$
\begin{array}{rcl}
T & ::= & \mathtt{r}?_{i\in I}\{l_i(x_i : S_i).T_i\} \;\mid\; \mathtt{r}!_{i\in I}\{l_i(x_i : S_i).T_i\} \\
& \mid & T \parallel T \;\mid\; T \oplus T \;\mid\; \mu \mathtt{t}.T \;\mid\; \mathtt{t} \;\mid\; \mathtt{end} \\
& \mid & \mathtt{call}\ \mathcal{P} : G\ \mathtt{with}\ (\widetilde{v}\ \mathtt{as}\ \widetilde{y} : \widetilde{S})\&(\widetilde{\mathtt{r}^2}).T \\
& \mid & \mathtt{ent}\ \mathcal{P}[\mathtt{r}]\langle\widetilde{v}\rangle\ \mathtt{from}\ \mathtt{r}.T \;\mid\; \mathtt{req}\ \mathcal{P}[\mathtt{r}]\langle\widetilde{v}\rangle\ \mathtt{to}\ \mathtt{r}.T
\end{array}
$$

- Trois opérateurs pour `calls`:
  - $\mathtt{call}\ \mathcal{P} : G\ \mathtt{with}\ (\widetilde{v}\ \mathtt{as}\ \widetilde{y} : \widetilde{S})\&(\widetilde{\mathtt{r}^{ext}})$ pour l'initiateur de la sous-session,
    - $\widetilde{v}$ instantient $\widetilde{y}$, $\widetilde{\mathtt{r}^{ext}}$ sont les roles invité de manière externe .
  - $\mathtt{req}\ \mathcal{P}[\mathtt{r}]\langle\widetilde{v}\rangle\ \mathtt{to}\ \mathtt{r}$ pour l'initiateur qui invite de manière interne.
  - $\mathtt{ent}\ \mathcal{P}[\mathtt{r}]\langle\widetilde{v}\rangle\ \mathtt{from}\ \mathtt{r}$: pour accepter une invitation interne.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Exemple: Projection de `calls`

$$
\begin{array}{l}
G_{CMS} = \texttt{let } \mathit{Contact} = \ldots \\
\texttt{in} \\
\text{client} \to \text{middle} : \text{request}(\mathit{req}_0). \\
\qquad (\text{middle} \to \text{client} : \text{answer}(\mathit{ans}_0).\texttt{end}) \\
\bigoplus^{\text{middle}} \\
\qquad (\text{middle } \texttt{calls } \mathit{Contact}(\text{middle}, \mathit{req}_0). \\
\qquad \text{middle} \to \text{client} : \text{answer}(\mathit{ans}_0).\texttt{end})
\end{array}
$$

- middle s'invite lui même dans *Contact*.

$$
\begin{array}{lcl}
G_{CMS} \Downarrow^{\emptyset}_{\text{client}} & = & \text{middle}!\{\text{request}(\mathit{req}_0)\}.\text{middle}?\{\text{answer}(\mathit{ans}_0)\} \\
G_{CMS} \Downarrow^{\emptyset}_{\text{middlew}} & = & \text{client}?\{\text{request}(\mathit{req}_0)\}. \\
 & & \quad \text{client}!\{\text{answer}(\mathit{ans}_0)\} \\
 & & \quad \oplus \\
 & & \quad (\texttt{call } \mathit{Contact} : G_C \texttt{ with } (\mathit{req}_0 \texttt{ as } \mathit{req} : \texttt{Req})\&(\text{server}). \\
 & & \qquad (\texttt{req } \mathit{Contact}[\text{agent}]\langle \mathit{req} \rangle \texttt{ to middle} \parallel \\
 & & \qquad \texttt{ent } \mathit{Contact}[\text{agent}]\langle \mathit{req} \rangle \texttt{ from middle} \parallel \\
 & & \qquad \text{client}!\{\text{answer}(\mathit{ans}_0)\}))
\end{array}
$$

Session-calcul

$$
\begin{array}{lcl}
P & ::= & \mathbf{0} \mid P|P \mid a(x).P \mid \overline{a}\langle s\rangle.P \mid P+P \\
& \mid & k?[\mathtt{r},\mathtt{r}]_{i\in I}\{l_i(x_i).P_i\} \mid k![\mathtt{r},\mathtt{r}]l\langle v\rangle.P \mid (\nu u)\ P \\
& \mid & \mathtt{new}\ s\ \mathtt{on}\ s\ \mathtt{with}\ (\widetilde{v})\&(\widetilde{a}\ \mathtt{as}\ \widetilde{\mathtt{r}}).P \\
& \mid & s\downarrow[\mathtt{r},\mathtt{r}:\mathtt{r}](x).P \mid s\uparrow[\mathtt{r},\mathtt{r}:\mathtt{r}]\langle s\rangle.P \mid \mu X(x).P\langle v\rangle \mid X\langle v\rangle
\end{array}
$$

- invitations externes sur a, internes sur s.
- 3 nouveaux opérateurs:

$$
\frac{\widetilde{\mathtt{r}}^2=(\mathtt{r}_1^2,\ldots,\mathtt{r}_n^2) \qquad \widetilde{a}=(a_1,\ldots,a_n)}{\mathtt{new}\ s\ \mathtt{on}\ k\ \mathtt{with}\ (\widetilde{v})\&(\widetilde{a}\ \mathtt{as}\ \widetilde{\mathtt{r}}^2).P \longrightarrow P \mid \overline{a_1}\langle s[\mathtt{r}_1^2]\rangle \mid \ldots \mid \overline{a_n}\langle s[\mathtt{r}_n^2]\rangle}
$$

$$
\frac{}{s\uparrow[\mathtt{r},\mathtt{r}':\mathtt{r}'']\langle k\rangle.P \mid s\downarrow[\mathtt{r},\mathtt{r}':\mathtt{r}''](x).Q \longrightarrow P \mid Q\{k/x\}}
$$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Validation

- Règle pour `new`:

$$\frac{\Gamma \vdash P \triangleright \Delta, s[\mathtt{r}] : T, k[\mathtt{r}_1^1]^\circ : T'_1, \ldots, k^\circ[\mathtt{r}_n^1] : T'_n, k^\bullet[\mathtt{r}_1^2] : T'_{n+1}, \ldots, k^\bullet[\mathtt{r}_m^2] : T'_{n+m} \quad \Gamma(\mathcal{P}) = (\widetilde{\mathtt{r}}^1, \widetilde{y}; \widetilde{\mathtt{r}}^2; G) \quad \forall i, \Gamma(a_i) = T'_{i+n}[\mathtt{r}_{i+n}] \quad \forall i, G\{\widetilde{v}/\widetilde{y}\} \Downarrow_{\mathtt{r}_i^1} = T'_i \quad \forall j, G\{\widetilde{v}/\widetilde{y}\} \Downarrow_{\mathtt{r}_j^2} = T'_{j+n} \quad \vdash \widetilde{v} : S \quad \Gamma(k) : \mathcal{P}\{\widetilde{v}/\widetilde{y}\}}{\Gamma \vdash \mathtt{new}\ k\ \mathtt{on}\ s\ \mathtt{with}\ (\widetilde{v})\&(\widetilde{a}\ \mathtt{as}\ \widetilde{\mathtt{r}}^2).P \triangleright \Delta, s[\mathtt{r}] : \mathtt{call}\ \mathcal{P} : G\ \mathtt{with}\ (\widetilde{v}\ \mathtt{as}\ \widetilde{y} : \widetilde{S})\&(\widetilde{\mathtt{r}}^2).T}$$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Session calculus: Exemple

- Un processus pour le protocole CMS:

$$
\begin{aligned}
P_{\text{alice}} = \quad & a(x).x![\text{client}, \text{middle}]\text{request}(\text{“kettle”}).x?[\text{middle}, \text{client}]\text{answer}(ans_0) \\
P_{\text{bob}} = \quad & \overline{a}\langle s\rangle.s?[\text{client}, \text{middle}]\text{request}(req_0). \\
& \quad (s![\text{middle}, \text{client}]\text{answer}(ans_0) \\
& + (\texttt{new } k \texttt{ on } s \texttt{ with } (req_0)\&(c \texttt{ as server}). \\
& \quad s \uparrow [\text{middle}, \text{middle} : \text{agent}]\langle k\rangle \mid s \downarrow [\text{middle}, \text{middle} : \text{agent}](z). \\
& \quad z![\text{agent}, \text{server}]\text{request}\langle req_0\rangle.z?[\text{server}, \text{agent}]\text{answer}(ans_r). \\
& \quad s![\text{middle}, \text{client}]\text{answer}\langle ans_r\rangle) \\
P_{\text{carol}} = \quad & c(y).y?[\text{agent}, \text{server}]\text{request}(req).y![\text{server}, \text{agent}]\text{answer}(ans)
\end{aligned}
$$

- carol est invitée (sur c) seulement si bob prend la seconde branche.
- Les règles de typage donnent:
  - $\Gamma \vdash P_{\text{alice}} \triangleright \emptyset$
  - $\Gamma \vdash P_{\text{bob}} \triangleright s[\text{middle}] : T_{\text{middle}}, s[\text{client}]^{\bullet} : T_{\text{client}}$
  - $\Gamma \vdash P_{\text{carol}} \triangleright \emptyset$

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Propriétés

- (Correspondence) Si $\Gamma \vdash P \triangleright \Delta$ alors $\mathrm{sat}(P, \Delta)_\Gamma$.
- (Subject Reduction) Si $\Gamma \vdash P \triangleright \Delta$ et $P \longrightarrow P'$ alors il existe Δ' t.q. $\Gamma \vdash P' \triangleright \Delta'$.
- (Progress) Si P est *unblocked* et $\Gamma \vdash P \triangleright \Delta$ t.q. Δ est *simple*, alors il existe P' t.q. $P \longrightarrow^+ P'$, $\Gamma \vdash P' \triangleright \Delta'$ et Δ' est cohérent.

- $\mathrm{sat}(P, \Delta)_\Gamma$ est une équivalence sémantique.
- P est *unblocked* quand il est $\neq \mathbf{0}$ et quand ses canaux partagés ne gênent pas les communications.
- Δ est simple s'il contient une seule session.
- Δ est cohérent s'il peut être obtenu depuis des types globaux.

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Implémentation en Scribble

```
import Req from <xsd> Req.xsd
import Dat from <xsd> Dat.xsd

global protocol Contact<sig request, sig answer>(role Question, role Server) {
    request(Req) from Question to Server;
    answer(Data) from Server to Question;
}

global protocol Main(role Client, role Middleware) {
    request(Req) from Client to Middleware;
    choice at Middleware {
        answer(Data) from Middleware to Question;
    } or {
        Middleware spawns Contact<forward,reply>(Middleware as Question);
        answer(Data) from Middleware to Question;
    }
}
```