6. The π-Calculus

June 21-29, 2022

The π-Calculus – Syntax

Let $\mathcal{N}$ be a set of names.

For names $x, y, z \in \mathcal{N}$, a **prefix** is an expression π of the form

$$\pi \quad ::= \quad \overline{x}\langle y\rangle \;\Big|\; x(z) \;\Big|\; [x = y]\pi \;\Big|\; \tau.$$

The set of all process expressions of $\mathcal{P}^{\pi}$ (the π-calculus) is defined by the following grammar:

$$P \quad ::= \quad \textstyle\sum_{i\in I} \pi_i.P_i \;\Big|\; P_1 \parallel P_2 \;\Big|\; (\nu a)P \;\Big|\; !P$$

The π-Calculus – Structural Congruence

Structural congruence $\equiv$ is the smallest process congruence on $\mathcal{P}^\pi$, such that

1. $[x = x]\pi.P \equiv \pi.P$;
2. $P \equiv_\alpha Q$ (α-conversion) implies $P \equiv Q$;
3. $P + \mathbf{0} \equiv P$, $P + Q \equiv Q + P$, $P + (Q + R) \equiv (P + Q) + R$;
4. $P \parallel \mathbf{0} \equiv P$, $P \parallel Q \equiv Q \parallel P$, $P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R$;
5. $(\nu x)(P \parallel Q) \equiv P \parallel (\nu x)Q$ if $x \notin \mathsf{fn}(P)$, $(\nu x)\mathbf{0} \equiv \mathbf{0}$, $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$;
6. $!P \equiv P \| !P$.

Note: Case 2, α-conversion, is often assumed as the default case, meaning that processes P and Q are not distinguished if $P \equiv_\alpha Q$ holds. We refrain from doing so for the procedure of this class and keep α-conversion inside structural congruence.

The π-Calculus – Reduction Semantics

The reduction relation for $\mathcal{P}^\pi$ is the smallest relation $\longrightarrow \subseteq \mathcal{P}^\pi \times \mathcal{P}^\pi$, satisfying the following rules:

$$\text{(tau)}\ \frac{}{\tau.P \longrightarrow P} \qquad \text{(struct)}\ \frac{P' \equiv P \quad P \longrightarrow Q \quad Q \equiv Q'}{P' \longrightarrow Q'}$$

$$\text{(react)}\ \frac{}{(\overline{x}\langle y\rangle.P_1 + M) \parallel (x(z).P_2 + N) \longrightarrow P_1 \parallel P_2\{{}^{y}/_{z}\}}$$

$$\text{(par)}\ \frac{P \longrightarrow P'}{P \parallel Q \longrightarrow P' \parallel Q} \qquad \text{(res)}\ \frac{P \longrightarrow P'}{(\nu x)P \longrightarrow (\nu x)P'}$$

Mobility – Scope Extrusion

$$\mathcal{Q} = (\nu z)(\overline{x}\langle z\rangle.P \parallel R) \parallel x(y).Q$$

with $z \notin \mathsf{fn}(P) \cup \mathsf{fn}(Q)$.

Then $\mathcal{Q} \longrightarrow P \parallel (\nu z)(R \parallel Q\{^z/_y\})$ since

1. $(\overline{x}\langle z\rangle.P) \parallel (x(y).Q) \longrightarrow P \parallel Q\{^z/_y\}$ due to (react) and (struct);
2. $(\overline{x}\langle z\rangle.P) \parallel (x(y).Q) \parallel R \longrightarrow P \parallel Q\{^z/_y\} \parallel R$ due to 1 and (par);
3. $\overline{x}\langle z\rangle.P \parallel R \parallel x(y).Q \longrightarrow P \parallel R \parallel Q\{^z/_y\}$ due to 2 and (struct);
4. $(\nu z)(\overline{x}\langle z\rangle.P \parallel R \parallel x(y).Q) \longrightarrow (\nu z)(P \parallel R \parallel Q\{^z/_y\})$ due to 3 and (res);
5. $(\nu z)(\overline{x}\langle z\rangle.P \parallel R) \parallel x(y).Q \longrightarrow P \parallel (\nu z)(R \parallel Q\{^z/_y\})$ due to 4 and (struct).

Such a behavior is also called **scope extrusion**.

The Polyadic π-Calculus

Every name $n \in \mathcal{N}$ has an arity $ar(n) \in \mathbb{N}$. A **polyadic input prefix** is an expression $x(y_1, \ldots, y_k)$ where $ar(x) = k$. A **polyadic output prefix** is an expression $\overline{x}\langle z_1, \ldots, z_k \rangle$ where $ar(x) = k$.

The **polyadic π-calculus** $\mathcal{P}^{\pi}_{\text{poly}}$ is the π-calculus using polyadic input/output prefixes. The reduction semantics is lifted to account for polyadic reactions.

Encoding $\mathcal{P}^{\pi}_{\text{poly}} \mapsto \mathcal{P}^{\pi}$:

1. $x(z_1, \ldots, z_{ar(x)}).P \mapsto x(z_1).x(z_2). \cdots .x(z_{ar(x)}).P'$ and $\overline{x}(y_1, \ldots, y_{ar(x)}).Q \mapsto \overline{x}\langle y_1 \rangle. \cdots .\overline{x}\langle y_{ar(x)} \rangle.Q'$ (where P' and Q' are likewise translated processes)

The Polyadic π-Calculus

Every name $n \in \mathcal{N}$ has an arity $ar(n) \in \mathbb{N}$. A **polyadic input prefix** is an expression $x(y_1, \ldots, y_k)$ where $ar(x) = k$. A **polyadic output prefix** is an expression $\overline{x}\langle z_1, \ldots, z_k \rangle$ where $ar(x) = k$.

The **polyadic π-calculus** $\mathcal{P}^{\pi}_{\text{poly}}$ is the π-calculus using polyadic input/output prefixes. The reduction semantics is lifted to account for polyadic reactions.

Encoding $\mathcal{P}^{\pi}_{\text{poly}} \mapsto \mathcal{P}^{\pi}$:

1. $x(z_1, \ldots, z_{ar(x)}).P \mapsto x(z_1).x(z_2). \cdots .x(z_{ar(x)}).P'$ and
   $\overline{x}(y_1, \ldots, y_{ar(x)}).Q \mapsto \overline{x}\langle y_1 \rangle. \cdots .\overline{x}\langle y_{ar(x)} \rangle.Q'$
   (where P' and Q' are likewise translated processes)
2. $x(z_1, \ldots, z_{ar(x)}).P \mapsto x(w).w(z_1). \cdots .w(z_{ar(x)}).Q'$ and
   $\overline{x}\langle y_1, \ldots, y_{ar(x)} \rangle \mapsto (\nu a)(\overline{x}(a).\overline{a}\langle y_1 \rangle. \cdots .\overline{a}\langle y_{ar(x)} \rangle.Q')$

π-Calculus with Process Calls

Additional processes to the ones in $\mathcal{P}^{\pi}_{\text{poly}}$ are process constants $A\langle\vec{x}\rangle$. Such a process constant comes with a defining equation $A(\vec{y}) := Q_A$, for which

$$Q_A = \cdots A\langle\vec{u}\rangle \cdots A\langle\vec{v}\rangle \cdots$$

and A may be called within a process

$$P = \cdots A\langle\vec{w}\rangle \cdots A\langle\vec{z}\rangle \cdots$$

Encoding Process Calls in $\mathcal{P}^{\pi}_{\text{poly}}$:

1. invent new name call_A for each process constant A;
2. in every process R, replace $A\langle\vec{w}\rangle$ by $\overline{\mathit{call}_A}$, yielding $\widehat{R}$;
3. replace the definition of P by

$$\widehat{\widehat{P}} = (\nu \mathit{call}_A)(\widehat{P} \| !\mathit{call}_A(\vec{x}).\widehat{Q_A})$$

Visible Actions

The set of π-**calculus actions** is given by

$$\pi \quad ::= \quad \overline{x}y \;\Big|\; xy \;\Big|\; \overline{x}(z) \;\Big|\; \tau$$

where $x, y, z \in \mathcal{N}$.

Free Output: represented by action $\pi = \overline{x}y$, where x is the so-called **subject of** π ($\mathsf{subj}(\pi) = x$), y its **object** ($\mathsf{obj}(\pi) = y$), $\mathsf{fn}(\pi) = \{x, y\}$, $\mathsf{bn}(\pi) = \emptyset$, $\mathsf{n}(\pi) = \{x, y\}$, $\pi\sigma = \overline{x\sigma}y\sigma$.

Input: $\pi = xy$, where $\mathsf{subj}(\pi) = x$, $\mathsf{obj}(\pi) = y$, $\mathsf{fn}(\pi) = \{x, y\}$, $\mathsf{bn}(\pi) = \emptyset$, $\mathsf{n}(\pi) = \{x, y\}$, and $\pi\sigma = x\sigma y\sigma$.

Bound Output: $\pi = \overline{x}(z)$, where $\mathsf{subj}(\pi) = x$, $\mathsf{obj}(\pi) = z$, $\mathsf{fn}(\pi) = \{x\}$, $\mathsf{bn}(\pi) = \{z\}$, $\mathsf{n}(\pi) = \{x, y\}$, and $\pi\sigma = \overline{x\sigma}(z)$.

Let us denote the set of all π-Calculus actions by $\mathcal{A}^{\pi}$.

LTS Semantics of the π-Calculus

$\mathcal{P}^\pi$ defines an LTS $(\mathcal{P}^\pi, \mathcal{A}^\pi, \rightarrow)$ where $\rightarrow$ is the smallest transition relation, satisfying the following rules.

$$\text{(out)}\ \frac{}{\overline{x}\langle y\rangle.P \xrightarrow{\overline{x}y} P} \qquad \text{(inp)}\ \frac{}{x(z).P \xrightarrow{xy} P\{y/z\}} \qquad \text{(tau)}\ \frac{}{\tau.P \xrightarrow{\tau} P}$$

$$\text{(mat)}\ \frac{\pi.P \xrightarrow{\alpha} P'}{[x = x]\pi.P \xrightarrow{\alpha} P'} \qquad \text{(sum-l)}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \text{(sum-r)}\ \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

$$\text{(par-l)}\ \frac{P \xrightarrow{\alpha} P' \quad \mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \emptyset}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \qquad \text{(par-r)}\ \frac{Q \xrightarrow{\alpha} Q' \quad \mathsf{bn}(\alpha) \cap \mathsf{fn}(P) = \emptyset}{P \parallel Q \xrightarrow{\alpha} P \parallel Q'}$$

$$\text{(comm-l)}\ \frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{xy} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \qquad \text{(comm-r)}\ \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\overline{x}y} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$$

LTS Semantics of the π-Calculus (cont'd)

$$\text{(close-l)}\ \frac{P \xrightarrow{\overline{x}(z)} P' \qquad Q \xrightarrow{xz} Q' \qquad z \notin \mathsf{fn}(Q)}{P \parallel Q \xrightarrow{\tau} (\nu z)(P' \parallel Q')} \qquad \text{(close-r)}\ \frac{\dots}{\dots}$$

$$\text{(res)}\ \frac{P \xrightarrow{\alpha} P' \qquad z \notin \mathsf{n}(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'} \qquad \text{(open)}\ \frac{P \xrightarrow{\overline{x}z} P' \qquad x \neq z}{(\nu z)P \xrightarrow{\overline{x}(z)} P'}$$

$$\text{(rep-act)}\ \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \| !P} \qquad \text{(rep-comm)}\ \frac{P \xrightarrow{\overline{x}y} P' \qquad P \xrightarrow{xy} P''}{!P \xrightarrow{\tau} (P' \parallel P'') \| !P}$$

$$\text{(rep-close)}\ \frac{P \xrightarrow{\overline{x}(z)} P' \qquad P \xrightarrow{xz} P'' \qquad z \notin \mathsf{fn}(P)}{!P \xrightarrow{\tau} (\nu z)(P' \parallel P'') \| !P}$$

$$\text{(alpha)}\ \frac{P \equiv_\alpha P' \qquad P \xrightarrow{\alpha} Q \qquad Q \equiv_\alpha Q'}{P' \xrightarrow{\alpha} Q'}$$

Properties of LTS

Theorem 6.1: The LTS $(\mathcal{P}^\pi, \mathcal{A}^\pi, \rightarrow)$ is image-finite

The following result is known as the **Harmony Lemma**:

Theorem 6.2: (1) $P \equiv\xrightarrow{\alpha} P'$ implies $P \xrightarrow{\alpha}\equiv P'$. (2) $P \longrightarrow P'$ if, and only if, $P \xrightarrow{\tau}\equiv P'$.

Proof Structure: For (1), we show that $Q \equiv R$ and $Q \xrightarrow{\alpha} Q'$ implies there is an R' with $R \xrightarrow{\alpha} R'$ and $Q' \equiv R'$.

For (2) and ($\Rightarrow$), $P \longrightarrow P'$ implies a standard form. For (2) and ($\Leftarrow$), argue by the inference rules for $P \xrightarrow{\tau} P'$ that $P \longrightarrow P'$.

Observations in the π-Calculus

Definition 6.3: For each name or co-name μ, define the **observability predicate** $\downarrow_\mu$ by

1. $P \downarrow_x$ if $P \xrightarrow{xy}$ for some $y \in \mathcal{N}$;
2. $P \downarrow_{\overline{x}}$ if $P \xrightarrow{\overline{x}y}$ or $P \xrightarrow{\overline{x}(z)}$ for some $y, z \in \mathcal{N}$.

Definition 6.4: Strong barbed bisimilarity is the largest symmetric relation $\sim^\bullet$, such that $P \sim^\bullet Q$ implies

1. $P \downarrow_\mu$ implies $Q \downarrow_\mu$ and
2. $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} \sim^\bullet P'$.

Strong barbed congruence is the largest relation $\sim^c \subseteq \sim^\bullet$, such that $P \sim^c Q$ implies $C[P] \sim^\bullet C[Q]$ for each context $C[\cdot]$.

Theorem 6.5: $P \sim^c Q$ if, and only if, for all substitutions σ and processes R, $P\sigma \parallel R \sim^\bullet Q\sigma \parallel R$.

The Asynchronous π-Calculus

The **asynchronous π-calculus** $\mathcal{P}^\pi_a$ is the following fragment of $\mathcal{P}^\pi$:

$$P \quad ::= \quad \overline{x}\langle y\rangle.\mathbf{0} \;\Big|\; M \;\Big|\; P \parallel P' \;\Big|\; (\nu z)P \;\Big|\; !P$$

$$M \quad ::= \quad \mathbf{0} \;\Big|\; x(z).P \;\Big|\; \tau.P \;\Big|\; M + M'$$

Definition 6.6: Asynchronous barbed bisimilarity is the largest symmetric process relation $\sim^\bullet_a$, such that $P \sim^\bullet_a Q$ implies

1. $P \downarrow_{\overline{x}}$ implies $Q \Downarrow_{\overline{x}}$ and
2. $P \xrightarrow{\tau} P'$ implies $Q \Rightarrow\sim^\bullet_a P'$.

Asynchronous barbed congruence is the largest relation $\sim^c_a \subseteq \sim^\bullet_a$, such that $P \sim^c_a Q$ implies $C[P] \sim^\bullet_a C[Q]$ for each process context C.