Introduction to Expressivenes in Concurrency

Frank D. Valencia
CNRS-LIX Ecole Polytechnique

Nov-Dec. 2007/MPRI

Motivation: The Notion of Expressiveness

Is the model $\mathcal{M}'$ as expressive as the model $\mathcal{M}$, written $\mathcal{M}' \succeq \mathcal{M}$?

- In Automata Theory: $\mathcal{M}' \succeq \mathcal{M}$ iff there exists a $f : \mathcal{M} \to \mathcal{M}'$ s.t. for each $M \in \mathcal{M}$, $\mathcal{L}(f(M)) = \mathcal{L}(M)$.
- E.g. $TM \succ PDS \succ FSA$ and the Chomsky Hierarchy: $UG \succ CSG \succ CFG \succ RG$
- The notion of expressiveness is well-understood and settled in automata theory.

Motivation: Expressiveness in Process Calculi

Is the calculus $\mathcal{C}'$ as expressive as the calculus $\mathcal{C}$, written $\mathcal{C}' \succeq \mathcal{C}$?

- In Concurrency Theory there is no yet an agreement upon expressiveness. In particular, there is no "Church-Turing Thesis" for Concurrency Theory.
- Intuitively $\mathcal{C}' \succeq \mathcal{C}$ iff for all $P \in \mathcal{C}$, there exists an encoding $[\![P]\!] \in \mathcal{C}'$ of P satisfying some correcteness criteria–e.g, preservation of behavioral equivalence: $P \sim [\![P]\!]$.

Motivation: Relevance of expressiveness studies

Many of the expressiveness studies Concurrency Theory resemble those for Logic, Formal Grammars, Distributed Computating. They involve:

- Identifying minimal set of operators for a given calculus. E.g., Is match/summation redundant in the π-calculus ?
- Identifying minimal terms forms for a given calculus. E.g., Is the asynchronous/monadic π-calculus as expressive as the synchronous/polyadic π-calculus ?
- Identifying meaningul decidable fragments of a given calculus. E.g., Is barbed equivalence decidable for CCS with replication ?
- Identifying problems a given calculus cannot solve. E.g., Can the asynchronous π calculus solve the leader election problem.
- Comparing conceptually different calculi. E.g., Can Ambients be encoded in the π-calculus ?

Outline

The π-calculus

The π-calculus (fragment) given in previous lectures:

Syntax:

P, Q	$::=$	$\mathbf{0}$	nil
	$\mid$	$P \parallel Q$	parallel composition of P and Q
	$\mid$	$\overline{c}\langle v\rangle.P$	output v on channel c and resume as P
	$\mid$	$c(x).P$	input from channel c
	$\mid$	$(\boldsymbol{\nu} x)P$	new channel name creation
	$\mid$	$!P$	replication

Free names (alpha-conversion follows accordingly):

$$\begin{array}{rclcrcl}
\mathrm{fn}(\mathbf{0}) & = & \emptyset & & \mathrm{fn}(P \parallel Q) & = & \mathrm{fn}(P) \cup \mathrm{fn}(Q) \\
\mathrm{fn}(\overline{c}\langle v\rangle.P) & = & \{c, v\} \cup \mathrm{fn}(P) & & \mathrm{fn}(c(x).P) & = & (\mathrm{fn}(P) \setminus \{x\}) \cup \{c\} \\
\mathrm{fn}((\boldsymbol{\nu} x)P) & = & \mathrm{fn}(P) \setminus \{x\} & & \mathrm{fn}(!P) & = & \mathrm{fn}(P)
\end{array}$$

Sometimes we use $P \mid Q$ and $\overline{c}v.P$ for $P \parallel Q$ and $\overline{c}\langle v\rangle.P$.

The π-calculus

Reduction relation

Structural congruence:

$$\begin{array}{rclcrcl} P \parallel 0 & \equiv & P & \qquad & P \parallel Q & \equiv & Q \parallel P \\ (P \parallel Q) \parallel R & \equiv & P \parallel (Q \parallel R) & \qquad & !P & \equiv & P \parallel !P \end{array}$$

$$(\boldsymbol{\nu} x)(\boldsymbol{\nu} y)P \equiv (\boldsymbol{\nu} y)(\boldsymbol{\nu} x)P$$

$$P \parallel (\boldsymbol{\nu} x)Q \equiv (\boldsymbol{\nu} x)(P \parallel Q) \text{ if } x \notin \mathrm{fn}(P)$$

Reduction rules:

$$\text{REACT} \quad \overline{c}\langle v\rangle.P \parallel c(x).Q \longrightarrow P \parallel Q\{^v/_x\}$$

$$\text{PAR}\ \frac{P \longrightarrow P'}{P \parallel Q \longrightarrow P' \parallel Q} \qquad \text{RES}\ \frac{P \longrightarrow P'}{(\boldsymbol{\nu} x)P \longrightarrow (\boldsymbol{\nu} x)P'} \qquad \text{STRUCT}\ \frac{P \equiv P' \longrightarrow Q' \equiv Q}{P \longrightarrow Q}$$

The π-calculus

Early Transitions

$$\textsc{Out}\ \frac{}{\overline{x}y.\,P \xrightarrow{\overline{x}y} P} \qquad \textsc{Inp}\ \frac{}{x(z).\,P \xrightarrow{xy} P\{y/z\}}$$

$$\textsc{Comm-L}\ \frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{xy} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \textsc{Par-L}\ \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\ \mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \emptyset$$

$$\textsc{Close-L}\ \frac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{xz} Q'}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu} z\,(P' \mid Q')}\ z \notin \mathsf{fn}(Q)$$

$$\textsc{Res}\ \frac{P \xrightarrow{\alpha} P'}{\boldsymbol{\nu} z\, P \xrightarrow{\alpha} \boldsymbol{\nu} z\, P'}\ z \notin \mathsf{n}(\alpha) \qquad \textsc{Open}\ \frac{P \xrightarrow{\overline{x}z} P'}{\boldsymbol{\nu} z\, P \xrightarrow{\overline{x}(z)} P'}\ z \neq x$$

$$\textsc{Rep-Act}\ \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P} \qquad \textsc{Rep-comm}\ \frac{P \xrightarrow{\overline{x}y} P' \quad P \xrightarrow{xy} P''}{!P \xrightarrow{\tau} (P' \mid P'') \mid !P}$$

$$\textsc{Rep-close}\ \frac{P \xrightarrow{\overline{x}(z)} P' \quad P \xrightarrow{xz} P''}{!P \xrightarrow{\tau} (\boldsymbol{\nu} z\,(P' \mid P'')) \mid !P}\ z \notin \mathsf{fn}(P)$$

Barbed Equivalences

Recall that $P \downarrow_\mu$ ($\mu \in \{x, \overline{x}\}$) iff $\exists \vec{z}, y, Q, R$ such that $x \notin \vec{z}$ and $P \equiv (\nu \vec{z})(\pi.Q \parallel R)$ and $\pi = x(y)$ if $\mu = x$ else $\pi = \overline{x}\langle y \rangle$.
Also $P \Downarrow_\mu$ iff $\exists Q$, $P \longrightarrow^* Q$ and $Q \downarrow_\mu$.

Definition (Barbed Bisimilarity)

(1) R is a ***barbed simulation*** iff for every $(P, Q) \in R$:
- If $P \longrightarrow P'$ then $\exists Q'$: $Q \longrightarrow^* Q' \ \wedge \ (P', Q') \in R$.
- If $P \downarrow_\mu$ then $Q \Downarrow_\mu$.

(2) (Barbed Bisimilarity) $P \dot{\approx} Q$ iff there is R such that R and R^{-1} are barbed simulations and $(P, Q) \in R$.
(3) (Barbed Congruence) $P \cong^{c} Q$ iff $K[P] \dot{\approx} K[Q]$ for every K.

(Early) Bisimulation Equivalences

Definition (Bisimilarity)

(1) R is a ***(strong) simulation*** iff for every $(P, Q) \in R$:
- If $P \xrightarrow{\alpha} P'$ then $\exists Q'$: $Q \xrightarrow{\alpha} Q' \;\wedge\; (P', Q') \in R$.

(2) (Strong Bisimilarity) $P \sim Q$ iff there is R such that R and R^{-1} are simulations and $(P, Q) \in R$.

(3) (Strong Full Bisimilarity) $P \sim^{c} Q$ iff $P\sigma \sim Q\sigma$ for every substitution σ.

The weak versions $\approx$ and $\approx^{c}$ are obtained by replacing $Q \xrightarrow{\alpha} Q'$ with $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ where $\stackrel{\hat{\alpha}}{\Longrightarrow}$ is $\xrightarrow{\tau}^{*}\xrightarrow{\alpha}\xrightarrow{\tau}^{*}$ if $\alpha \neq \tau$, and $\xrightarrow{\tau}^{*}$ otherwise.

Encodings

Encoding

An encoding $[\![\cdot]\!] : \mathcal{C} \rightarrow \mathcal{C}'$ is a map from $\mathcal{C}$ to $\mathcal{C}'$. The encoding of $P \in \mathcal{C}$ is denoted as $[\![P]\!]$.

Encodings: $[\![\cdot]\!] : \pi^2 \longrightarrow \pi$

Recall the encoding of the bi-adic π-calculus (π^2) into π.

Example

[Milner 91] The encoding $[\![\cdot]\!] : \pi^2 \longrightarrow \pi$ is defined as

$$\begin{array}{rcl} [\![\overline{x}\langle z_1, z_2\rangle.P]\!] & = & (\nu w)\overline{x}\langle w\rangle.\overline{w}\langle z_1\rangle.\overline{w}\langle z_2\rangle.[\![P]\!] \\ [\![x(y_1, y_2).Q]\!] & = & x(w).w(y_1).w(y_2).[\![Q]\!] \end{array}$$

$[\![\cdot]\!] : \pi^2 \longrightarrow \pi$ is a homomorphism for the other cases.

- In what sense is $[\![\cdot]\!] : \pi^2 \longrightarrow \pi$ correct ?
- Question: How about the encoding from asynchronous π ($A\pi$) into π ?

Encodings: $[\![\cdot]\!] : \pi \longrightarrow A\pi$

Definition (Synchronous into asynchronous)

[Boudol 92] The encoding $[\![\cdot]\!] : \pi \longrightarrow A\pi$ is defined as

$$\begin{array}{rcl} [\![\overline{x}\langle z\rangle.P]\!] & = & (\nu w)(\overline{x}\langle w\rangle \parallel w(u).(\overline{u}\langle z\rangle \parallel [\![P]\!])) \\ [\![x(y).Q]\!] & = & x(w).(\nu u)(\overline{w}\langle u\rangle \parallel u(y).[\![Q]\!]) \end{array}$$

$[\![\cdot]\!] : A\pi \longrightarrow \pi$ is a homomorphism for the other cases.

- How about using a protocol of two exchanges only ?

Two steps protocol

[Honda-Tokoro 92]. The encoding $[\![\cdot]\!] : \pi \longrightarrow A\pi$ is defined as

$$\begin{array}{rcl} [\![\overline{x}\langle z\rangle.P]\!] & = & x(w).(\overline{w}\langle z\rangle \parallel [\![P]\!]) \\ [\![x(y).Q]\!] & = & (\nu w)(\overline{x}\langle w\rangle \parallel w(y).[\![Q]\!]) \end{array}$$

Encodings: $[\![\cdot]\!] : K\pi \longrightarrow \pi$

- $K\pi$ extends π with finitely many paremetric recursive definitions: $P := \ldots \mid K\langle\vec{z}\rangle$
- Each $K\langle\vec{z}\rangle$ has a unique $K(\vec{y}) \stackrel{\text{def}}{=} P$ with $|\vec{z}| = |\vec{y}|$.
- Transition rule: (Cons) $K\langle\vec{z}\rangle \xrightarrow{\tau} P\{\vec{z}/\vec{y}\}$ if $K(\vec{y}) \stackrel{\text{def}}{=} P$.
- Let $K^1\pi$ be $K\pi$ but with a single monadic definition.

Definition (Encoding of $K^1\pi$)

[Milner 91] The encoding $[\![\cdot]\!] : K^1\pi \longrightarrow \pi$ is defined as $[\![P]\!] = (\nu k)([\![P]\!]_0 \parallel [\![K(y) \stackrel{\text{def}}{=} P]\!]_0)$ where

$$\begin{array}{lcl} [\![K\langle z\rangle]\!]_0 & = & \overline{k}\langle z\rangle \\ [\![K(y) \stackrel{\text{def}}{=} P]\!]_0 & = & !k(w).[\![P]\!]_0 \end{array}$$

$[\![\cdot]\!]_0$ is a homomorphism for the other cases.

Expressiveness Criteria

Correctness Criteria

In what sense are the above encodings "correct" ?

The most commonly used criteria/requirenment for correctness of the encodings are:

- Preservation of Behavioral Equivalence.
- Preservation of Observations.
- Operational Correspondence.
- Full Abstraction.
- Structural Requirements: Compositionality and Homomorphisms.

Expressiveness Criteria: Preservation of Equivalence

Semantic Preservation wrt $\bowtie$

$\forall P \in \mathcal{C}$, we must have $[\![P]\!] \ \bowtie \ P$.

- Typically $\bowtie$ is some bisimilarity relation.
- Natural and it could be a very strong correspondence depending on the chosen $\bowtie$.
- But it presupposes that the source and taget calculi are equipped with $\bowtie$.
- $[\![\cdot]\!] : \pi^2 \rightarrow \pi$ satisfies the above with $\bowtie = \ \dot{\approx}$ but not for $\bowtie = \ \cong^c$.
- $[\![\cdot]\!] : K^1\pi \rightarrow \pi$ satisfies the above with $\bowtie = \ \cong^c$.

Expressiveness Criteria: Preservation of Observables

Preservation of Observations

$\forall P \in \mathcal{C}$, we must have $\text{obs}([\![P]\!]) = \text{obs}(P)$.

Here obs(.) denotes a set of observations than can be made of processes in $\mathcal{C} \cup \mathcal{C}'$: Typically barbs, traces, divergence, test, failures.

- Observations such as barbs and traces are not enough to capture process behaviour.
- Failures are often enough.
- $[\![\cdot]\!] : \pi^2 \rightarrow \pi$ satisfies the above for barbs but not for tests.
- $[\![\cdot]\!] : K^1\pi \rightarrow \pi$ satisfies the above for barbs and tests.

Expressiveness Criteria: Operational Correspondence

Operational correspondence

$\forall P, Q \in \mathcal{C}$, (a) If $P \longrightarrow Q$ then $[\![P]\!] \longrightarrow^{*} \bowtie\ [\![Q]\!]$ and
(b) $\forall R$ if $[\![P]\!] \longrightarrow R$ then $\exists R'$ s.t. $P \longrightarrow R'$ and $R \bowtie [\![R']\!]$.

- (a) Preservation of reduction steps (Soundness).
- (b) Reflexion of reduction steps (Completeness).
- It conveys the notion of operational simulation.
- Significant aspects are not covered (e.g., some observables)
- $[\![\cdot]\!] : \pi^2 \rightarrow \pi$ satisfies the above for $\bowtie = \cong^c$.
- $[\![\cdot]\!] : K^1\pi \rightarrow \pi$ satisfies the above for $\bowtie = \cong^c$ and for label transitions.

Expressiveness Criteria: Full Abstraction

Full Abstraction

$\forall P, Q \in \mathcal{C},\ P \ \bowtie_{\mathcal{C}}\ Q$ if and only if $[\![P]\!] \ \bowtie_{\mathcal{C}'}\ [\![Q]\!]$.

I.e. equivalent processes are mapped into equivalent processes.

- If Direction: Soundness.
- Only-If Direction: Completeness.
- Useful when $[\![P]\!]$ and P cannot be compared directly.
- Completeness could be too demanding if $\bowtie$ is a congruence.
- $[\![\cdot]\!] : \pi^2 \rightarrow \pi$ is fully abstract sound but not complete for $\bowtie = \ \cong^c$.
- $[\![\cdot]\!] : K^1\pi \rightarrow \pi$ is fully abstract $\bowtie = \ \cong^c$.

Expressiveness Criteria: Weak Full Abstraction

Weak Full Abstraction

$\forall P, Q \in \mathcal{C}$,
$K[P] \bowtie_{\mathcal{C}} K[Q]$ for all $\mathcal{C}$-context K
if and only if
$[\![K]\!][\,[\![P]\!]\,] \bowtie_{\mathcal{C}'} [\![K]\!][\,[\![Q]\!]\,]$ for all $\mathcal{C}$-context K.

Here $\bowtie$ is typically a non-congruence like barbed bisimulation, trace equivalence, etc.

- Completeness wrt "encoded contexts".
- $[\![\cdot]\!] : \pi^2 \to \pi$ is weakly fully abstract for $\bowtie = \dot{\approx}$.

Expressiveness Criteria: Compositionality

Compositionality and Homomorphism

(1) The encoding $[\![\cdot]\!] : \mathcal{C} \rightarrow \mathcal{C}'$ is compositional wrt an n-ary operator op if and only if there exists a $\mathcal{C}'$-context K with n-holes such that $[\![op(P_1, \ldots, P_n)]\!] = K[[\![P_1]\!], \ldots, [\![P_n]\!]]$.
(2) $[\![\cdot]\!] : \mathcal{C} \rightarrow \mathcal{C}'$ is weakly compositional iff $\exists K, \forall P \; [\![P]\!] = K[\; [\![P]\!]' \;]$ where $[\![\cdot]\!]'$ is compositional.
(3) $[\![\cdot]\!] : \mathcal{C} \rightarrow \mathcal{C}'$ is homomorphic wrt an n-ary operator op in $\mathcal{C}$ if and only if $[\![op(P_1, \ldots, P_n)]\!] = op([\![P_1]\!], \ldots, [\![P_n]\!])$.

- Homomorphism is sometimes required for the parallel operator: $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$.
- Compositionality and its weak version are often required.
- $[\![\cdot]\!] : \pi^2 \rightarrow \pi$ is compositional for all the operators.
- $[\![\cdot]\!] : K^1\pi \rightarrow \pi$ is not compositional but weakly compositional.

Correctness of $[\![\cdot]\!] : K^1\pi \rightarrow \pi$.

Let $[\![\cdot]\!] : K^1\pi \rightarrow \pi$ be the encoding from $K\pi$ with a single monadic recursive definitions into π.

Theorem (Operational Correspondence)

(1) If $P \xrightarrow{\alpha} Q$ then $[\![P]\!] \xrightarrow{\alpha} \sim [\![Q]\!]$
(2) If $[\![P]\!] \xrightarrow{\alpha} R$ then $\exists Q\ P \longrightarrow Q$ and $R \sim [\![Q]\!]$.

Proof.

(1) and (2) proceed by induction on the inference and on the size of processes using the Replication Theorem. □

Theorem (Replication Theorem (Sangiorgi's Book))

If x occurs in P_i $(i \in I)$ and R only in output subject position then $(\nu x)(\prod_{i\in I} P_i \parallel !x(y).R) \sim^c \prod_{i\in I}(\nu x)(P_i \parallel !x(y).R)$.

Correctness of $[\![\cdot]\!] : K^1\pi \longrightarrow \pi$.

Theorem (Semantic Preservation wrt $\sim^c$)

$P \sim^c [\![P]\!]$

Proof.

Verify that $\mathcal{R} = \{(P, [\![P]\!])\}$ is a bisimulation up-to $\sim$ using the Operational Correspondence. Also $\mathcal{R}$ is closed under substitutions. $\square$

Theorem (Full Abstraction)

$P \cong^c Q$ *iff* $[\![P]\!] \cong^c [\![Q]\!]$.

Proof.

Since $\sim^c \; = \; \cong^c$ and the Semantic preservation wrt $\sim^c$. $\square$

Correctness of $[\![\cdot]\!] : \pi^2 \to \pi$.

Let $[\![\cdot]\!] : \pi^2 \to \pi$ be the encoding from bi-adic π to π.

Theorem (Operational Correspondence)

(1) if $P \longrightarrow Q$ then $[\![P]\!] \longrightarrow^* [\![Q]\!]$ and
(2) If $[\![P]\!] \longrightarrow R$ then $\exists Q;\ P \longrightarrow Q$ and $R \cong^c [\![Q]\!]$.

The proof of (1) is by induction on the inference. The proof (2) is rather involved because arbitrary application of $\equiv$ in $[\![P]\!] \longrightarrow R$.

Theorem (preservation of barbs)

$P \downarrow_\mu$ iff $[\![P]\!] \downarrow_\mu$

Theorem (Semantic preservation wrt $\dot{\approx}$)

$[\![P]\!] \dot{\approx} P$.

Correctness of $[\![\cdot]\!] : \pi^2 \longrightarrow \pi$.

Corollary (Soundness)

If $[\![P]\!] \cong^c [\![Q]\!]$ then $P \cong^c Q$.

Proof.

From the homomorphic definition of $[\![\cdot]\!]$ and the preservation of $\dot{\approx}$.
$K[P] \dot{\approx} [\![K[P]]\!] = [\![K]\!][\ [\![P]\!]\ ] \dot{\approx} [\![K]\!][\ [\![Q]\!]\ ] = [\![K[Q]]\!] \dot{\approx} C[Q]$ $\square$

Correctness of $[\![\cdot]\!] : \pi^2 \longrightarrow \pi$.

Corollary (Soundness)

If $[\![P]\!] \cong^c [\![Q]\!]$ then $P \cong^c Q$.

Exercises :

- Show that the encoding is not complete. I.e., $P \cong^c Q$ does not imply $[\![P]\!] \cong^c [\![Q]\!]$.
- Are the encodings $[\![\cdot]\!] : A\pi \longrightarrow \pi$ by Boudol and Honda complete wrt $\cong^c$? If not, prove it.
- Define a weakly compositional encoding $[\![\cdot]\!] : K\pi \longrightarrow \pi$ which is sound wrt $\cong^c$? Is your encoding complete $\cong^c$? If not, argue why.

Open Question: Is there a compositional encoding $[\![\cdot]\!] : \pi^2 \longrightarrow \pi$ fully-abstract wrt $\cong^c$.

Trios

A trios process is a polyadic π process whose prefixes are of the form $\pi'.\pi.\pi''.0$. Trios processes can encode arbitrary polyadic π processes [Parrow'01].

Exercise: Give an encoding $[\![\cdot]\!]$ from π^0 processes into π^0 trios processes. Argue that $[\![P]\!] \approx P$.

Replication vs Recursion in CCS

Notice that π^0 is CCS with replication instead of recursive definitions $CCS_!$.

- Is $CCS_!$ as expressive as CCS? We shall conclude this section we a survey on these kind of Recursion vs Replication results.

References

- Kohei Honda, Mario Tokoro: *An Object Calculus for Asynchronous Communication*. ECOOP 1991: 133-147. 1991.
- Gerard Boudol: *Asynchrony and the π-calculus.* Rapport de Recherche RR-1702, INRIA-Sophia Antipolis. 1992
- Robin Milner: *The Polyadic pi-Calculus: A Tutorial.* Technical Report LFCS report ECS-LFCS-91-180, University of Edinburgh. 1994.
- J. Aranda, C. Di Giusto, C. Palamidessi and F. Valencia. *On Recursion, Replication and Scope Mechanisms in Process Calculi*. To appear in FMCO'06. ©Springer-Verlag. 2007.

Computational Expressiveness of CCS

Language of a process :

$$L(P) = \{s \in \mathcal{L}^* | \ \exists Q : P \stackrel{s}{\Longrightarrow} Q \wedge \forall \alpha \in Act : Q \stackrel{\alpha}{\not\longrightarrow}\}.$$

Theorem (CCS can generate CFL)

For any context-free grammar G, there exists a CCS process P_G such that $s \in L(G)$ iff $s \in L(P_G)$.

Proof.

Hint: Consider productions in Chomsky Normal form: $A \longrightarrow B.C$ or $A \longrightarrow a$. For the case $B.C$ provide a definition $A(\ldots) \stackrel{\texttt{def}}{=} \ldots.$ which allows for the sequentialization of B and C. $\square$

Computational Expressiveness of CCS

Theorem

CCS is Turing-Expressive.

This can be shown by encoding Minsky Machines.

Minsky's Two-Counter Machines

Sequence of labelled instructions on two counters c_0 and c_1:

$$\begin{array}{lcl} L_i & : & \texttt{halt} \\ L_i & : & c_n := c_n + 1; \texttt{goto}\ L_j \\ L_i & : & \texttt{if}\ c_n = 0\ \texttt{then}\ \texttt{goto}\ L_j\ \texttt{else}\ c_n := c_n - 1;\ \texttt{goto}\, L_k \end{array}$$

The machine: 1) starts at L_1, 2) halts if control reaches the location of a halt instruction and 3) computes the value n if it halts with $c_0 = n$.

Computational Expressiveness of CCS

Definition (A Counter C)

$$C \stackrel{\text{def}}{=} isz.C + inc.(\nu l)(C'\langle l\rangle \parallel l.C)$$
$$C'(l) \stackrel{\text{def}}{=} dec.\bar{l}.0 + inc.(\nu l')(C'\langle l'\rangle \parallel l'.C'\langle l\rangle)$$

For counters X and Y replace C with X, resp Y, and isz, inc, dec with $iszX, incX, decX$, resp $iszY, incY, decY$.

Instructions are represented as processes waiting for an input on its label.

Example

L_2 : $\texttt{if}\ \ X = 0\ \ \texttt{then}\ \ \texttt{goto}\ \ L_4\ \texttt{else}\ \ \texttt{goto}\ L_8$ and L_4 : *halt* can be represented as $L_2 \stackrel{\text{def}}{=} l_2.(\overline{iszX}.\bar{l}_4.L_2 + \overline{decX}.\overline{incX}.\bar{l}_8.L_2)$ and $L_4 \stackrel{\text{def}}{=} l_4.\overline{halt}$.

Computational Expressiveness of CCS

Definition (A program M)

A program $M(X,Y) = L_1 : I_1; \ldots ; L_n : I_n$ can be encoded as

$$[\![M(X,Y)]\!] = (\nu l_1 \ldots l_n)(\bar{l}_1.0 \parallel L_1 \parallel \ldots \parallel L_n \parallel X \parallel Y)$$

The correctness is stated as follows:

Theorem (Correctness)

$M(X,Y)$ computes n on X if and only if

$$([\![M]\!] \parallel halt.Dec_n) \Downarrow_{\overline{yes}}$$

where for $n > 0$, $Dec_n = \overline{decX}.Dec_{n-1}$ and $Dec_0 = \overline{iszX}.\overline{yes}$

Computational Expressiveness of CCS $\pi^0 = CCS_!$

Theorem (π^0 can generate REG)

Given a regular expression e, there exists a CCS! process P_e such that $s \in L(e)$ iff $s \in L(P_e)$.

Exercise. Write a CCS! process P such that $L(P) = a^* c$.

Computational Expressiveness of CCS $\pi^0 = CCS_!$

Theorem (π^0 can generate REG)

Given a regular expression e, there exists a CCS process P_e such that $s \in L(e)$ iff $s \in L(P_e)$.

Proof.

Definition 4. *Given a regular expression e, we define $[\![e]\!]$ as the $CCS_!$ process $(\nu\, m)([\![e]\!]_m \mid m)$ where $[\![e]\!]_m$, with $m \notin fn([\![e]\!])$, is inductively defined as follows:*

$$
\begin{array}{ll}
[\![\emptyset]\!]_m = & DIV \\
[\![\epsilon]\!]_m = & \overline{m} \\
[\![a]\!]_m = & a.\overline{m} \\
[\![e_1 + e_2]\!]_m = & \begin{cases} [\![e_1]\!]_m & \text{if } L(e_2) = \emptyset \\ [\![e_2]\!]_m & \text{if } L(e_1) = \emptyset \\ [\![e_1]\!]_m + [\![e_2]\!]_m & \text{otherwise} \end{cases} \\
[\![e_1.e_2]\!]_m = & (\nu\, m_1)([\![e_1]\!]_{m_1} \mid m_1.[\![e_2]\!]_m) \text{ with } m_1 \notin fn(e_1) \\
[\![e^\star]\!]_m = & \begin{cases} \overline{m} & \text{if } L(e) = \emptyset \\ (\nu\, m')(\overline{m'} \mid !m'.[\![e]\!]_{m'} \mid m'.\overline{m}) \text{ with } m' \notin fn(e) & \text{otherwise} \end{cases}
\end{array}
$$

Computational Expressiveness of CCS $\pi^0 = CCS_!$

Theorem (π^0 can generate REG)

Given a regular expression e, there exists a CCS! process P_e such that $s \in L(e)$ iff $s \in L(P_e)$.

But CCS! can generate CFL languages too.

Exercise. Write a CCS! process Q such that $L(Q) = a^n b^n$.
Hint: Recall the process P such that $L(P) = a^n c$.

Computational Expressiveness of $\pi^0 = CCS_!$

- CCS! is also Turing Expressive: It can also encode Minsky Machines.
- The encoding is unfaithfull: $[\![M]\!]$ can evolve into a process which does NOT correspond to any computation of M.
  - Such process however never terminates (i.e., it is divergent).
- In fact, CCS! cannot encode even CFG faithfully.
  - The following theorem and $a^n b^n c$ are central to this impossibility result:

Theorem

Let $P \in CCS!$. Suppose that $P \stackrel{s.\alpha}{\Longrightarrow}$ where $s \in Act^*$. Then $P \stackrel{s'.\alpha}{\Longrightarrow}$ for some $s' \in Act^*$ whose length is bounded by a value depending only on the size of P.

Computational Expressiveness of $\pi^0 = CCS_!$

The construction in CCS! differs for registers.

Counter in CCS!

$$
\begin{array}{lll}
C & \stackrel{\text{def}}{=} \overline{c} \parallel & !c.(\nu m, i, d, u)(\overline{m} \parallel !m.(inc.\overline{i} + dec.\overline{d}) \parallel \\
& & !i.(\overline{m} \parallel \overline{inc'} \parallel \overline{u} \parallel d.u.(\overline{m} \parallel \overline{dec'}) \parallel \\
& & d.(\overline{isz} \parallel u.DIV \parallel \overline{c})\ )
\end{array}
$$

Instructions: $L_2 : \texttt{if}\ \ X = 0\ \texttt{then goto}\ L_4\,\texttt{else goto}\,L_8$ can be modelled as

$$!l_2.\overline{decX}.(dec'X.\overline{l_4} + iszX.\overline{l_8})$$

Computational Expressiveness of $\pi^0 = CCS_!$

Theorem (Correctness)

$M(X, Y)$ computes n on X if and only if

$$\exists Q : (\llbracket M \rrbracket \parallel halt.Dec_n) \Longrightarrow (Q \parallel \overline{yes}) \land Q \not\longrightarrow$$

where for $n > 0$ $Dec_n = \overline{decX}.dec'X.Dec_{n-1}$ *and* $D_0 = iszX.\overline{yes}$.

References

- N. Busi, M.. Gabbrielli, G. Zavattaro: *Comparing Recursion, Replication, and Iteration in Process Calculi*. ICALP 2004: 307-319.
- N. Busi, M.. Gabbrielli, G. Zavattaro: *Replication vs. Recursive Definitions in Channel Based Calculi*. ICALP 2003: 133-144.
- J. Aranda, C. Di Giusto, M. Nielsen and F. Valencia. *CCS with Replication in the Chomsky Hierarchy: The Expressive Power of Divergence*. APLAS'07.

Linearity.

Linearity of messages and input processes.

- In the π-calculus outputs (messages) and inputs are ***linear***.
- E.g. the parallel composition

$$\bar{x}z \mid x(y).P \mid x(w).Q$$

reduces either

- to

$$P\{z/y\} \mid Q$$

- or to

$$P \mid Q\{z/w\}$$

Persistence.

Persistence of messages.

- Other calculi follow a different pattern: *Messages are **persistent***. E.g.:
- ***Concurrent Constraint Programming*** (CCP)[Saraswat'90] where

  information can only increase during computation.

- Several ***Calculi for Security*** (e.g., Winskel&Crazolara's SPL) to model a Dolev-Yao assumption:

  "The Spy sees and remembers every message in transit"

Persistence.

Persistence of messages and input process.

- Persistent π input processes model functions, procedures and higher-order communication (also arises in the notion of ω-receptiveness) [Sangiorgi'99].
- Persistent messages and input processes can be used to reason about protocols that can run unboundedly (see [Blanchet'04]).

Linearity vs Persistence.

- Does the persistence assumption restrict the kind of systems that can be reasoned about ? E.g.
- Can some security attacks based on linear messages be impossible to model under the persistent message assumption of SPL?
- Is **Linear** CCP more expressive than CCP ?

Linearity vs Persistence.

To study the expressiveness of fragments of π capturing the above sources of persistence:

- $A\pi$: *Asynch. π-calculus*, here denoted simply as π:

$$P, Q \;:=\; \bar{x}z \;\mid\; x(y).P \;\mid\; P \mid Q \;\mid\; (\nu x)P \;\mid\; !P$$

- $PO\pi$: *Persistent-output* (messages) π:

$$P, Q \;:=\; !\bar{x}z \;\mid\; x(y).P \;\mid\; P \mid Q \;\mid\; (\nu x)P \;\mid\; !P$$

- $PI\pi$: *Persistent-input* π:

$$P, Q \;:=\; \bar{x}z \;\mid\; !x(y).P \;\mid\; P \mid Q \;\mid\; (\nu x)P \;\mid\; !P$$

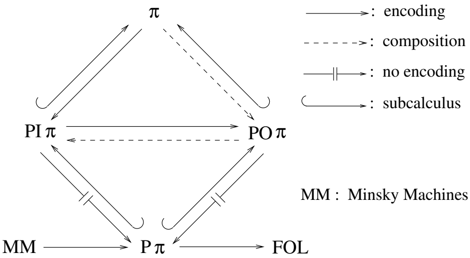
- $P\pi$: *Persistent* (input & ouput) π:

$$P, Q \;:=\; !\bar{x}z \;\mid\; !x(y).P \;\mid\; P \mid Q \;\mid\; (\nu x)P \;\mid\; !P$$

Encodings & Interpretations

Some studies on Linearity vs Persistence have reported:

- The (non) existance of, *compositional encodings* between the fragments *fully-abstract* wrt barbed congruence and barbed bisimilarity.
- The *Turing-Completeness* of $P\pi$.
- A *compositional FOL* interpretation of $P\pi$.



Applications: Decidability

As applications of the above and classic FOL results there are

- Decidability results of *barbed-congruence* for *n*-adic versions.

	$P\pi$	$PO\pi$	$PI\pi$
0	yes	yes	no
1	?	no	
2	no		

- Identify meaningful decidable *infinite-state* mobile classes of π processes.

Impossibility of a sound encoding of π in $P\pi$

Impossibility of Sound Encodings

There is no encoding $[\![\cdot]\!] : \pi \rightarrow P\pi$, homomorphic wrt parallel composition, such that $[\![P]\!] \cong^c [\![Q]\!]$ implies $P \cong^c Q$.

- $\cong^c$ is barbed congruence for $P\pi$ (the result also holds for barbed bisimilarity)
- **Key property**: in $P\pi$, for every P, $P \mid P \cong^c P$.
- Key property is not trivial for $P = (\nu x)Q$ and it does not hold with mismatch. Notice that $R = !x(y).!x(y').[y \neq y'].t$ distinguishes $Q = (\nu z)!\bar{x}z$ from $Q \mid Q$.

Impossibility of a sound encoding of π in $P\pi$

Theorem

There is no encoding $[\![\cdot]\!] : \pi \rightarrow P\pi$, homomorphic wrt parallel composition, such that $[\![P]\!] \cong^c [\![Q]\!]$ implies $P \cong^c Q$.

The proof involves the following lemmas:

1. If $P \longrightarrow Q$ then $P\sigma \longrightarrow Q\sigma$.
   - Does it hold with mistmatch?
2. $P \cong^c Q$ iff $\forall R, \sigma : \; P\sigma \parallel R \,\dot{\approx}\, Q\sigma \parallel R$. (Textbook)
3. $P \,\dot{\approx}\, Q$ iff $P \,\overset{\circ}{\simeq}\, Q$ where
   - $\dot{\approx}$ is barbed bisimilarity for $P\pi$ and
   - $P \,\overset{\circ}{\simeq}\, Q$ holds iff $P \Downarrow_{\overline{x}} \iff Q \Downarrow_{\overline{x}}$)
4. $P \parallel P \cong^c P$.

Take $P = Q \parallel Q$, $Q = \overline{x} \parallel x.x.\overline{t}$. Notice that $P \not\cong^c Q$. But from (4) $[\![P]\!] = [\![Q]\!] \parallel [\![Q]\!] \cong^c [\![Q]\!]$. It remains to prove (3) and (4).

FOL Characterization of $P\pi$

FOL Interpretation of $P\pi$

$$[\![!\bar{x}z]\!] = out(x,z), \quad [\![!x(y).P]\!] = \forall_y \; out(x,y) \Rightarrow [\![P]\!]$$
$$[\![(\nu x)P]\!] = \exists_x [\![P]\!], \quad [\![P \mid Q]\!] = [\![P]\!] \wedge [\![Q]\!].$$

- *Input* and *"new"* binders are interpreted as *universal* and *existential* quantifiers.

Theorem: FOL Characterization of Barbed Observability

$$[\![P]\!] \models \exists_z out(x,z) \text{ if and only if } P \Downarrow_{\bar{x}}$$

- ***With mismatch***and $[\![x \neq y.P]\!] = x \neq y \Rightarrow [\![P]\!]$, $Q = (\nu y)(\nu y')[y \neq y'].!\bar{x}z \Downarrow_{\bar{x}}$ but $[\![Q]\!] \not\models \exists_z out(x,z)$.

FOL Characterization of $P\pi$

- Consider the following example:
- In $P = !x(y).Q \mid (\nu z)!xz$, by extruding the private name z, we can conclude that $(\nu z)Q\{z/y\}$ is *executed* in P.
- In $[\![P]\!] = \forall_y out(x, y) \Rightarrow [\![Q]\!] \ \wedge \ \exists_z out(x, z)$, by moving the existential z to outermost position, we conclude that $\exists_z [\![Q]\!]\{z/y\}$ is a *logical consequence* of $[\![P]\!]$.
- FOL interpretation captures name extrusion ($P\pi$-calculus mobility) in FOL via *existential* and *universal* quantifiers.

Encoding $A\pi$ in the semi-persistent calculi

Consider $S = \bar{x}u \mid \bar{x}w \mid x(y).\bar{y}m \mid x(y).\bar{y}n$. Want an encoding $[\![S]\!]$ in the semi-persistent calculi s.t.:

- $[\![S]\!] = [\![\overline{x}\langle u\rangle]\!] \mid [\![\overline{x}\langle w\rangle]\!] \mid [\![x(y).\bar{y}m]\!] \mid [\![x(y).\bar{y}n]\!]$ behaves
- either as (a)$[\![\overline{u}\langle m\rangle]\!] \mid [\![\overline{w}\langle n\rangle]\!]$ or as (b)$[\![\overline{w}\langle m\rangle]\!] \mid [\![\overline{u}\langle n\rangle]\!]$.
- ***Problem:*** In either case input and outputs are both consumed. In the semi-persistent calculi either input or outputs cannot be consumed.

Encoding $A\pi$ in $PO\pi$

- The encoding $[\![\cdot]\!] : \pi \rightarrow PI\pi$ is a homomorphism for all operators but:

$$[\![x(\vec{y}).P]\!] = (\nu\, t\, f)(\bar{t} \mid !x(\vec{y}).(\nu\, l\,)(\bar{l} \mid !t.!l.([\![P]\!] \mid !\bar{f}) \mid !f.!l.\bar{x}\langle\vec{y}\rangle))$$

- The idea is a suitable combination of locking and forwarding mechanisms: If the $[\![x(y).P]\!]$ has already received a message then it forwards the current message.
- **Key property:** In asynch. π, forwarders (e.g., $!x(y).\bar{x}y$) are barbed congruent to the null process 0.

Theorem

Every (asynch) π process P is (weak) barbed congruent to $[\![P]\!]$.

Encoding $A\pi$ in $PO\pi$

- Consider the encoding $[\![\cdot]\!] : \pi \rightarrow PO\pi$:

$$\begin{aligned} [\![\overline{x}\langle\vec{z}\rangle]\!] &= (\nu s)(!\overline{x}\langle s\rangle \mid s(r).!\overline{r}\langle\vec{z}\rangle) \\ [\![x(\vec{y}).P]\!] &= x(s).(\nu r)(!\overline{s}\langle r\rangle \mid r(\vec{y}).[\![P]\!]) \end{aligned}$$

- **Problem**: An encoded input may get deadlocked. E.g.,
  - Consider $[\![\overline{x}\langle u\rangle]\!] \mid [\![\overline{x}\langle w\rangle]\!] \mid [\![x(y).P]\!] \mid [\![x(y).Q]\!]$.
  - Suppose $[\![x(y).P]\!]$ gets the u of $[\![\overline{x}\langle u\rangle]\!]$. Then $[\![x(y).Q]\!]$ may input the broadcast s of $[\![\overline{x}\langle u\rangle]\!]$ and get stuck waiting on r ***unable to interact*** with $[\![\overline{x}\langle w\rangle]\!]$.
- But this wouldn't be a problem if inputs were ***persistent*** as in $PI\pi$: If a copy becomes unable to interact with, there is always another able to.
- **Solution:** Encode first π into $PI\pi$ and then compose the encodings.

References on Linearity vs Persistence

- C. Palamidessi, V. Saraswat, F. Valencia and B. Victor. *On the Expressiveness of Linearity vs Persistence in the Asynchronous Pi Calculus*. LICS 2006:59-68.
- D. Cacciagrano, F. Corradini, J. Aranda, F. Valencia. *Persistence and Testing Semantics in the Asynchronous Pi Calculus*. EXPRESS'07.

Expressive Power of Asynchronous Communication

Motivation: To understand the expressive power of $A\pi$.

- It's theory is simpler and somewhat more satisfactory.
- We've seen how it encodes (synchronous, polyadic) π. Recall e.g., Boudol's encoding.
- But for the encodings are for π *without summation.*
- We shall see how it encodes various forms of summation.
- We shall see it cannot encode arbitrary summation.
- We'll do this by using *electoral problems* solvable in π but not in $A\pi$.

Some Distintive Properties for $A\pi$

1. If $P \xrightarrow{\overline{x}\langle y\rangle} P'$ then $P \equiv \overline{x}\langle y\rangle \parallel P'$.
2. If $P \xrightarrow{\overline{x}\langle y\rangle} \xrightarrow{\alpha} P'$ then $P \xrightarrow{\alpha} \xrightarrow{\overline{x}\langle y\rangle} P' \equiv P'$.
3. If $P \xrightarrow{\overline{x}\langle y\rangle} \xrightarrow{xw} P'$ with $w \notin fn(P)$ then $P \xrightarrow{\tau} \equiv P'\{y/w\}$.

Exercise: Show (1-3) then Theorem below. Does (2) hold for $\Longrightarrow$?

Theorem (Diamond Property)

$$\begin{array}{ccccc} & & P & & \\ & \overline{x}y \swarrow & & \searrow xw & \\ Q & & & & R \\ & xw \searrow & & \swarrow \overline{x}y & \\ & & S & & \end{array}$$

Equivalences for $A\pi$

Definition (Asynchronous Barbed Bisimilarity)

(1) *Asynchronous barbed bisimilarity* is the largest symmetric relation $\dot{\approx}_a$ such that whenever $P \dot{\approx}_a Q$,

(a) $P \downarrow_{\overline{x}}$ implies $Q \Downarrow_{\overline{x}}$

(b) $P \xrightarrow{\tau} P'$ implies $Q \Rightarrow \dot{\approx}_a P'$.

(2) P and Q are *asynchronous barbed congruent*, $P \cong_a^c Q$, if $C[P] \dot{\approx}_a C[Q]$ for every context C of $A\pi$.

Definition (Asynchronous Bisimilarity)

Asynchronous bisimilarity is the largest symmetric relation, $\approx_a$, such that whenever $P \approx_a Q$,

(1) if $P \xrightarrow{\alpha} P'$ and α is $\overline{x}y$ or $\overline{x}(z)$ or τ, then $Q \stackrel{\hat{\alpha}}{\Longrightarrow} \approx_a P'$

(2) if $P \xrightarrow{xy} P'$ then (a) $Q \stackrel{xy}{\Longrightarrow} \approx_a P'$ or

(b) $Q \Rightarrow Q'$ and $P' \approx_a (Q' \mid \overline{x}y)$.

Equivalences for $A\pi$

Some useful properties in $A\pi$:

1. If $P \approx_{\mathrm{a}} Q$ then $P \cong^{\mathrm{c}}_{\mathrm{a}} Q$.
2. $x(y).\overline{x}\langle y \rangle \cong^{\mathrm{c}}_{\mathrm{a}} 0$ (I.e., forwarders are equivalent to 0).

   Exercise: Show (2).

The π calculus with prefixed summations

The π-calculus prefixed summation π^{Σ} extends the π fragment we've considered so far with guarded summations:

- $P := \ldots \mid \sum_{i \in I} \pi_i . P_i$

Reduction rule for summation

$$\text{R-INTER} \quad \frac{}{(\overline{x}y.\,P_1 + M_1) \mid (x(z).\,P_2 + M_2) \longrightarrow P_1 \mid P_2\{y/z\}}$$

Transition rule for summation

$$\text{SUM-L} \quad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$$

The π calculus with blind choice: $\pi^{\Sigma\tau}$.

In the blind-choice π-calculus, summation takes the form $\sum_{i \in I} \tau.P_i$.

Exercises:

1. Give an encoding $[\![\cdot]\!] : A\pi^{\Sigma\tau} \longrightarrow A\pi$ such that $[\![P]\!] \sim P$.
2. Show that there cannot be an encoding $[\![\cdot]\!] : A\pi^{\Sigma} \longrightarrow A\pi$ such that $[\![P]\!] \sim P$.

The π calculus with input-choice: $\pi^{\Sigma i}$

In input-choice π summations takes the form $\sum_{i \in I} x_i(y_i).P_i$.

Encoding into Asynchronous (polyadic) π

$$[\![\Sigma_i\, x_i(z).\, P_i]\!] \stackrel{\text{def}}{=} \nu\ell(\ \mathrm{PROCEED}(\ell) \\ \mid \Pi_i\, x_i(z).\, (\nu p, f)\ (\overline{\ell}\langle p, f\rangle \mid p.\, (\mathrm{FAIL}(\ell) \mid [\![P_i]\!]) \\ \mid f.\, (\mathrm{FAIL}(\ell) \mid \overline{x_i}\langle z\rangle)))$$

where ℓ, p, and f are fresh and

$$\mathrm{PROCEED}(\ell) \stackrel{\text{def}}{=} \ell(p, f).\overline{p}$$
$$\mathrm{FAIL}(\ell) \stackrel{\text{def}}{=} \ell(p, f).\overline{f}\ .$$

Exercises:

1. Let $\mathcal{E}$ be the above encoding. Show that $\exists P : \mathcal{E}(P) \not\approx_a P$.
2. Give $\mathcal{E}' : A\pi^{\Sigma i} \rightarrow A\pi$ so that $\forall P : \mathcal{E}'(P) \approx_a P$.
3. Then show that $\mathcal{E}$ is neither sound nor complete.

The π calculus with input-choice: $\pi^{\Sigma i}$

Encoding into Asynchronous (polyadic) π

$$\begin{aligned}[\Sigma_i\, x_i(z).\, P_i] \stackrel{\text{def}}{=} \nu\ell(\ &\mathrm{PROCEED}(\ell)\\ &|\Pi_i\, x_i(z).\,(\nu p,f)\,(\overline{\ell}\langle p,f\rangle \mid p.\,(\mathrm{FAIL}(\ell) \mid [P_i])\\ &\qquad\qquad\qquad\qquad\qquad\quad \mid f.\,(\mathrm{FAIL}(\ell) \mid \overline{x_i}\langle z\rangle)))\end{aligned}$$

where ℓ, p, and f are fresh and

$$\begin{aligned}\mathrm{PROCEED}(\ell) &\stackrel{\text{def}}{=} \ell(p,f).\overline{p}\\ \mathrm{FAIL}(\ell) &\stackrel{\text{def}}{=} \ell(p,f).\overline{f}\,.\end{aligned}$$

Observations and Hints:

- Consider $P = \overline{x}\langle z\rangle \parallel x(y).\overline{y} + w(y).0$ to show $\mathcal{E}(P) \not\approx_a P$.
- Note that $\mathcal{E}$ and $\mathcal{E}'$ act as the identity on their images. So $\mathcal{E}(\mathcal{E}(P)) = \mathcal{E}(P)$ and $\mathcal{E}(\mathcal{E}'(P)) = \mathcal{E}'(P)$.
- However $\mathcal{E}(P) \bowtie P$ where $\bowtie \stackrel{\text{def}}{=}$ *coupled-bisimulation.*

The π calculus with separate-choice: $\pi^{\Sigma s}$

In separate-choice summation can be $\sum_i x_i(y_i).P_i$ or $\sum_i \overline{x_i}\langle y_i\rangle.P_i$

Encoding into Asynchronous (polyadic) π

$$
\begin{array}{l}
[\![\Sigma_i\, \overline{x_i} d_i.\, P_i]\!] \stackrel{\text{def}}{=} \nu s\ \ (\ \mathrm{PROCEED}\langle s\rangle \\
\qquad\qquad\qquad\quad \mid \Pi_i\ \nu a\, \overline{x_i}\langle d_i, s, a\rangle.\,(\nu p, f)\ (\overline{a}\langle p, f\rangle \mid p.\,[\![P_i]\!] \mid f.\,0)) \\
[\![\Sigma_i\, y_i(z).\, Q_i]\!] \stackrel{\text{def}}{=} \nu r\ \ (\ \mathrm{PROCEED}\langle r\rangle \\
\qquad\quad \mid \Pi_i\ \nu g\,(\ \overline{g} \\
\qquad\qquad\quad \mid !g.\, y_i(z, s, a).(\nu p_1, f_1)(\ \overline{r}\langle p_1, f_1\rangle \\
\qquad\qquad\qquad\qquad \mid p_1.\,(\nu p_2, f_2)\ \ (\ \overline{s}\langle p_2, f_2\rangle \\
\qquad\qquad\qquad\qquad\qquad \mid p_2.\ \ (\ \mathrm{FAIL}\langle r\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid \mathrm{FAIL}\langle s\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid \mathrm{PROCEED}\langle a\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid [\![Q_i]\!]) \\
\qquad\qquad\qquad\qquad\qquad \mid f_2.\ \ (\ \mathrm{PROCEED}\langle r\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid \mathrm{FAIL}\langle s\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid \mathrm{FAIL}\langle a\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid \overline{g})) \\
\qquad\qquad\qquad\qquad \mid f_1.\,(\mathrm{FAIL}\langle r\rangle \mid \overline{y_i}\langle z, s, a\rangle)))))
\end{array}
$$

The π calculus with mixed choice: π^{Σ}

In π^{Σ}summations are mixed. Can we encode them using the obvious generalization of the previous
encoding of $\pi^{\Sigma s}$?

- Consider $P = x_1(y).P_1 + \overline{x_2}\langle w\rangle.P_2 \ \parallel\ \overline{x_1}\langle w\rangle.Q_1 + x_2(y).Q_2$
- How about other encodings?

Impossibility Result

Under certain reasonable restrictions, no encoding of mixed-choice into $A\pi$ can exist.

Background: Hypergraphs

Definition (Hypergraphs)

A hypergraph is a tuple $H = \langle N, X, t\rangle$ where N, X are finite sets whose elements are called *nodes* and *edges* (or *hyperedges*) respectively, and t (*type*) is a function which assigns to each $x \in X$ a set of nodes, representing the nodes *connected* by x. We will also use the notation $x : n_1, \ldots, n_k$ to indicate $t(x) = \{n_1, \ldots, n_k\}$.

Definition (Automorphism)

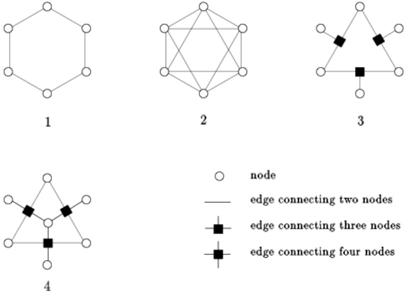
The concept of graph automorphism extends naturally to hypergraphs: Given a hypergraph $H = \langle N, X, t\rangle$, an *automorphism* on H is a pair $\sigma = \langle \sigma_N, \sigma_X\rangle$ such that $\sigma_N : N \to N$ and $\sigma_X : X \to X$ are permutations which preserve the type of edges, namely for each $x \in X$, if $x : n_1, \ldots, n_k$, then $\sigma_X(x) : \sigma_N(n_1), \ldots, \sigma_N(n_k)$.

- The *orbit* $n \in X$ by σ is $O_\sigma(n) = \{n, \sigma(n), \sigma^2(n), \ldots, \sigma^h(n)\}$ where h is the least power s.t. $\sigma^h = id$.

Background: Hypergraphs

- σ is *well-balanced* iff all of its orbits have the same cardinality.
- E.g., (1) and (2) have a one with a single orbit of size 6, (4) has none.

Examples



Networks

- A *(process) network* P of size k takes the form $P_1 \parallel \ldots \parallel P_k$.
- A *computation* C of the P takes the form:
- $$\begin{array}{lll} P_1|P_2|\ldots|P_k & \xrightarrow{\mu^0} & P_1^1|P_2^1|\ldots|P_k^1 \\ & \xrightarrow{\mu^1} & P_1^2|P_2^2|\ldots|P_k^2 \\ & \vdots & \\ & \xrightarrow{\mu^{n-1}} & P_1^n|P_2^n|\ldots|P_k^n \\ & (\xrightarrow{\mu^n} & \ldots) \end{array}$$
- $Proj(C, i)$ is the contributions of P_i to C: The sequence of transitions performed by P_i in C.
- $$P_i \xRightarrow{\vec{\mu}^0} P_i^1 \xRightarrow{\vec{\mu}^1} P_i^2 \xRightarrow{\vec{\mu}^2} \ldots \xRightarrow{\vec{\mu}^{n-1}} P_i^n \; (\xRightarrow{\vec{\mu}^n} \ldots)$$

Electoral Networks

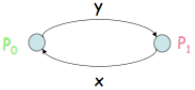
- A *(process) network* P of size k takes the form $P_1 \parallel \ldots \parallel P_k$.
- A *computation* C of the P takes the form:
- $$\begin{array}{rcl} P_1 | P_2 | \ldots | P_k & \xrightarrow{\mu^0} & P_1^1 | P_2^1 | \ldots | P_k^1 \\ & \xrightarrow{\mu^1} & P_1^2 | P_2^2 | \ldots | P_k^2 \\ & \vdots & \\ & \xrightarrow{\mu^{n-1}} & P_1^n | P_2^n | \ldots | P_k^n \\ & (\xrightarrow{\mu^n} & \ldots) \end{array}$$
- $Proj(C, i)$ is the contributions of P_i to C: The sequence of transitions performed by P_i in C.
- $$P_i \stackrel{\vec{\mu}^0}{\Longrightarrow} P_i^1 \stackrel{\vec{\mu}^1}{\Longrightarrow} P_i^2 \stackrel{\vec{\mu}^2}{\Longrightarrow} \ldots \stackrel{\vec{\mu}^{n-1}}{\Longrightarrow} P_i^n \; (\stackrel{\vec{\mu}^n}{\Longrightarrow} \ldots)$$

Electoral Networks in π

- A network $P = P_1 \parallel \ldots \parallel P_k$ is an *electoral system* iff for every computation C of P:
  - C can be extended to a computation C' and
  - $\exists n \leq k$ (the "leader") s.t.,
    - $\forall i \leq k$: $Proj(C', i)$ contains the action $\overline{out}n$, and
    - no extension of C' contains any action $\overline{out}m$ with $m \neq n$.
- The *hypergraph of* P, $H(P) = \langle N, X, t \rangle$ is given by:
  - $N = \{1, \ldots, k\}$,
  - $X = fn(P) - \{out\}$,
  - $t(x) = \{n \mid x \in fn(P_n) \}$

Symmetric Electoral Networks in π

- Given $P = P_1 \parallel \ldots \parallel P_k$, let σ be an automorphism on $H(P)$.
    - P is *symmetric* wrt σ iff for each $i \leq k$,
        - $P_{\sigma(i)} \equiv P_i\sigma$.
    - P is symmetric iff symmetric wrt all automorphism on $H(P)$
- Notice that P is symmetric wrt σ then it is symmetric wrt σ^i $(i > 1)$.
- Symmetric electoral system in π^Σ with ring structure:



$P_0 = x.\overline{out}\,0 + \bar{y}.\overline{out}\,1$

- $P_1 = y.\overline{out}\,1 + \bar{x}.\overline{out}\,0$

Symmetric Electoral Networks in $A\pi$.

Theorem (Impossibility of electoral systems)

Let $P = P_1 \parallel \ldots \parallel P_k$ be a $A\pi$ network so that $H(P)$ is a ring with $k > 1$. Assume that P is symmetric wrt σ where σ has a single orbit on $H(P)$. Then P cannot be an electoral system.

The proof strategy involves:

1. Building a computation $P \xrightarrow{\mu_1} P^1 \ldots \xrightarrow{\mu_h} P^h$ so that P^h is a symmetric network for every $h > 1$.
2. Usind Diamond Lemma and Symmetry of P^{h-1} to build P^h.
3. The symmetry cannot be broken, hence no leader can be selected.

Symmetric Electoral Networks in $A\pi$.

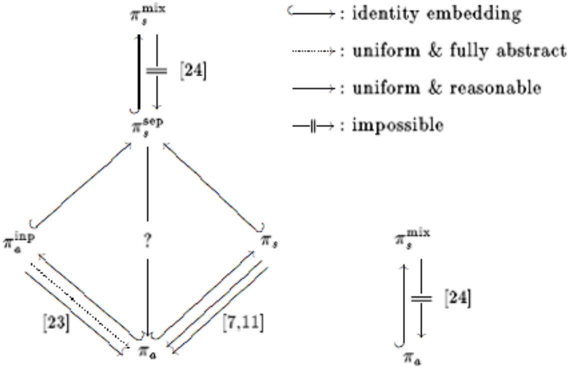
Corollary:

- There is no encoding $[\![\cdot]\!] : \pi^{\Sigma} \to A\pi$ such that
  1. $[\![P \parallel Q]\!] = [\![P]\!] \parallel [\![Q]\!]$
  2. $[\![P\sigma]\!] = [\![P]\!]\sigma$
  3. Preservation of obervables (actions on visible channels) on maximal computations.

Proof idea: (1) and (2) preserve symmetry and (3) distinguishes an electoral system from a non-electoral one.

Summary

Expressiveness Hierarchy



References

- Catuscia Palamidessi: *Comparing The Expressive Power Of The Synchronous And Asynchronous Pi-Calculi*. Mathematical Structures in Computer Science 13(5): 685-719 (2003).
- Uwe Nestmann: *What is a "Good" Encoding of Guarded Choice?* Inf. Comput. 156(1-2): 287-319 (2000).
- Uwe Nestmann, Benjamin C. Pierce: *Decoding Choice Encodings.* Inf. Comput. 163(1): 1-59 (2000)

Exercises: Non-Complete Encodings

Exercises :

- Show that the encoding $[\![\cdot]\!] : \pi^2 \ \longrightarrow \pi$ is not complete. I.e., $P \cong^c Q$ does not imply $[\![P]\!] \cong^c [\![Q]\!]$.
  - Take $P = \overline{x}\langle yz\rangle.0 \parallel \overline{x}\langle yz\rangle.0$ and $Q = \overline{x}\langle yz\rangle.\overline{x}\langle yz\rangle.0$. Consider the context $K = [\cdot] \parallel x(u).x(w).\overline{t}\langle t\rangle$.
- Are the encodings $[\![\cdot]\!] : A\pi \ \longrightarrow \pi$ by Boudol and Honda complete wrt $\cong^c$? If not, prove it.
  - Boudol's as above and Honda's as above but with $P = x(y).0 \parallel x(y).0$ and $Q = x(y).x(y).0$.
- Define a weakly compositional encoding $[\![\cdot]\!] : K\pi \ \longrightarrow \ \pi$ which is sound wrt $\cong^c$? Is your encoding complete $\cong^c$? If not, argue why.
  - Take the composite encoding $K\pi \rightarrow \pi^n \rightarrow \pi$. Notice that the polyadic communication occur on the private channels.

Exercises: Trios

A trios process is a polyadic π process whose prefixes are of the form $\pi'.\pi.\pi''.0$. Trios processes can encode arbitrary polyadic π processes [Parrow'01].

Exercise Give an encoding $[\![\cdot]\!]$ from π^0 processes into π^0 trios processes so that $[\![P]\!] \approx P$.

Exercises: Trios

A trios process is a polyadic π process whose prefixes are of the form $\pi'.\pi.\pi''.0$. Trios processes can encode arbitrary polyadic π processes [Parrow'01].

Exercise Give an encoding $[\![\cdot]\!]$ from π^0 processes into π^0 trios processes so that $[\![P]\!] \approx P$.

Solution

Definition 6. *Given a $CCS_!$ process P, $[\![P]\!]$ is the trios-process $(\nu\, l)(\tau.\tau.\bar{l} \mid [\![P]\!]_l)$ where $[\![P]\!]_l$, with $l \notin n(P)$, is inductively defined as follows:*

$$
\begin{array}{ll}
[\![0]\!]_l = & 0 \\
[\![\alpha.P]\!]_l = & (\nu\, l')(l.\alpha.\overline{l'} \mid [\![P]\!]_{l'}) \text{ where } l' \notin n(P) \\
[\![P \mid Q]\!]_l = & (\nu\, l', l'')(l.\overline{l'}.\overline{l''} \mid [\![P]\!]_{l'} \mid [\![P]\!]_{l''}) \text{ where } l', l'' \notin n(P) \cup n(Q) \\
[\![!P]\!]_l = & (\nu\, l')(!l.\overline{l'}.\bar{l} \mid ![\![P]\!]_{l'}) \text{ where } l' \notin n(P) \\
[\![(\nu\, x)P]\!]_l = & (\nu\, x)[\![P]\!]_l
\end{array}
$$

Exercises: Language of Processes

Exercises:

- Write a CCS! process P such that $L(P) = a^*c$.
  - $P = (\nu l)(\bar{l} \parallel !(l.a.\bar{l}) \parallel l.c)$
- Write a CCS! process Q such that $L(Q) = a^n b^n$.
  - $P = (\nu l)(\bar{l} \parallel !(l.a.(\bar{l} \parallel u)) \parallel l.!u.b)$

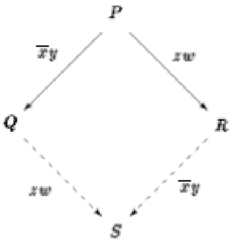
Exercises: Properties of $A\pi$

In $A\pi$ the following holds:

1. If $P \xrightarrow{\overline{x}\langle y\rangle} P'$ then $P \equiv \overline{x}\langle y\rangle \parallel P'$.
2. If $P \xrightarrow{\overline{x}\langle y\rangle} \xrightarrow{\alpha} P'$ then $P \xrightarrow{\alpha} \xrightarrow{\overline{x}\langle y\rangle} P' \equiv P'$.
3. $x(y).\overline{x}\langle y\rangle \cong^{c}{}_{a}\ 0$.

Exercise: Show (1) and (2) then Theorem below. Also show (3).

Theorem (Diamond Property for $A\pi$)



Exercises for Choice Operators.

In the blind-choice π-calculus, summation takes the form $\sum_{i \in I} \tau.P_i$.

Exercises:

1. Give an encoding $[\![\cdot]\!] : A\pi^{\Sigma\tau} \longrightarrow A\pi$ from asynchronous π with blind-choice to $A\pi$ such that $[\![P]\!] \sim P$.
2. Show that there cannot be an encoding $[\![\cdot]\!] : A\pi^{\Sigma} \longrightarrow A\pi$ from asynchronous π with choice to $A\pi$such that $[\![P]\!] \sim P$.

Exercises for Choice Operators

Encoding into Asynchronous (polyadic) π

$$
\begin{aligned}
[\Sigma_i\, x_i(z).\, P_i] \stackrel{\text{def}}{=} \boldsymbol{\nu}\ell(\ & \mathrm{PROCEED}(\ell) \\
& \mid \Pi_i\, x_i(z).\, (\boldsymbol{\nu} p, f)\ (\overline{\ell}\langle p, f\rangle \mid p.\, (\mathrm{FAIL}(\ell) \mid [P_i]) \\
& \qquad\qquad\qquad\qquad\qquad\quad \mid f.\, (\mathrm{FAIL}(\ell) \mid \overline{x_i}\langle z\rangle)))
\end{aligned}
$$

where ℓ, p, and f are fresh and

$$
\begin{aligned}
\mathrm{PROCEED}(\ell) &\stackrel{\text{def}}{=} \ell(p, f).\overline{p} \\
\mathrm{FAIL}(\ell) &\stackrel{\text{def}}{=} \ell(p, f).\overline{f}\ .
\end{aligned}
$$

Exercises:

1. Let $\mathcal{E}$ be the above encoding. Show that $\exists P : \mathcal{E}(P) \not\approx_a P$.
2. Give $\mathcal{E}' : A\pi^{\Sigma i} \longrightarrow A\pi$ so that $\forall P : \mathcal{E}'(P) \approx_a P$.
3. Then show that $\mathcal{E}$ is neither sound nor complete.
   - Hint: Consider $P = \overline{x}\langle z\rangle \parallel x(y).\overline{y} + w(y).0$ to show $\mathcal{E}(P) \not\approx_a P$.
   - Hint: Note that $\mathcal{E}$ acts as the identity on its images. So $\mathcal{E}(\mathcal{E}(P)) = \mathcal{E}(P)$ and $\mathcal{E}(\mathcal{E}'(P)) = \mathcal{E}'(P)$.