The pi-Calculus (Part 1)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.uni-linz.ac.at

The pi-Calculus

- Process calculus developed in continuation of the work on CCS.
  - Robin Milner, Joachim Parrow, David Walker. *A Calculus of Mobile Processes*. Information and Computation, 100:1–40, 1992.
  - Robin Milner. *Elements of Interaction*. Turing Award Lecture. Communications of the ACM, 36(1):78–89, January 1993.
  - Robin Milner. *The Polyadic π-calculus: a Tutorial*. F.L. Bauer et al (eds), Logic and Algebra of Specification, Springer 1993, pp. 203–246.
- Designed to capture mobility.
  - Concurrent systems whose configuration may change.
- Highly influential with many extensions and applications:
  - Abadi and Gordon (1997): Spi-calculus (cryptographic protocols).
  - Shapiro et al (2000): BioSPI (biological processes).
  - Formal modeling of web service architectures (WS-BPEL, ...).
  - Semantics of object-oriented languages.
  - ...

1. **CCS Revisited**

2. From CCS to the π-Calculus

3. The π-Calculus

A Reformulation of CCS

- Names $\{a, b, \ldots\}$ and Co-names $\{\bar{a}, \bar{b}, \ldots\}$
  - Complement $\bar{a}$ of a, $\bar{\bar{a}} = a$.
  - Labels $\{a, \bar{a}, b, \bar{b}, \ldots\}$
  - $\vec{a} = a_1, \ldots, a_n$
- Process Identifiers $\{A, B, \ldots\}$
  - Defining Equation $A(\vec{a}) := P_A$
    - P_A is a process expression whose free names are included in $\vec{a}$.
- Concurrent Process Expressions

$$P ::= A\langle a_1, \ldots, a_n\rangle \mid \sum_{i \in I} \alpha_i.P_i \mid P_1|P_2 \mid \text{new } a\ P$$

  - Summation $\sum_{i \in I} \alpha_i.P_i$ with finite indexing set I
    - $P_1 + P_2 + P_3 = \sum_{i \in \{1,2,3\}} .P_i$
    - $0 = \sum_{i \in \emptyset} .P_i$
  - Restriction new a P
    - Name a is bound (not free) in the restriction.

Structural Congruence

- Process Congruence: an equivalence relation $\simeq$ on concurrent process expressions is a *process congruence*, if $P \simeq Q$ implies
  - $\alpha.P + M \simeq \alpha.Q + M$
  - $\text{new } a\ P \ \simeq\ \text{new } a\ Q$
  - $P|R \simeq Q|R,\ R|P \simeq R|Q$
- Structural Congruence: the structural congruence $\equiv$ is the process congruence defined by the following equations:
  1. Change of bound names (alpha-conversion).
  2. Reordering of terms in a summation.
  3. $P|0 \equiv P,\ P|Q \equiv Q|P,\ P|(Q|R) \equiv (P|Q)|R$.
  4. $\text{new } a\ (P|Q) \equiv P|\text{new } a\ Q$, if a not free in P. $\text{new } a\ 0 \equiv 0, \text{new } a\ b\ P \equiv \text{new } b\ a\ P$.
  5. $A\langle\vec{b}\rangle \equiv \{\vec{b}/\vec{a}\}P_A$, if $A(\vec{a}) := P_A$.

Used in the definition of the possible process reactions.

Standard Forms

- Standard Form: a process expression

  $\text{new } \vec{a}\ (M_1 \mid \ldots \mid M_n)$
  - Each M_i is a non-empty sum.
  - If $n = 0$, the standard form is new $\vec{a}\ 0$.
  - If $\vec{a}$ is empty, the standard form is $M_1 \mid \ldots \mid M_n$.
- Theorem: Every process is structurally congruent to a standard form.

Reactions

- Reaction Relation $\rightarrow$: set of those transitions that can be inferred from the following rules:

TAU $\tau.P + M \rightarrow P$

REACT $(a.P + M)|(\bar{a}.Q + N) \rightarrow P|Q$

PAR $$\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q}$$

RES $$\frac{P \rightarrow P'}{\text{new } a\ P \rightarrow \text{new } a\ P'}$$

STRUCT $$\frac{P \rightarrow P'}{Q \rightarrow Q'}, \text{ if } P \equiv Q \text{ and } P' \equiv Q'$$

The internal reactions within a process.

Labelled Transitions

- Transition Relation $\stackrel{\alpha}{\rightarrow}$: set of transitions that can be inferred from the following rules (where α is either a label λ or τ):

SUM$_t$ $M + \alpha.P + N \stackrel{\alpha}{\rightarrow} P$

REACT$_t$ $$\frac{P \stackrel{\lambda}{\rightarrow} P' \quad Q \stackrel{\bar{\lambda}}{\rightarrow} Q'}{P|Q \stackrel{\tau}{\rightarrow} P'|Q'}$$

LPAR$_t$ $$\frac{P \stackrel{\alpha}{\rightarrow} P'}{P|Q \stackrel{\alpha}{\rightarrow} P'|Q}$$ RPAR$_t$ $$\frac{Q \stackrel{\alpha}{\rightarrow} Q'}{P|Q \stackrel{\alpha}{\rightarrow} P|Q'}$$

RES$_t$ $$\frac{P \stackrel{\alpha}{\rightarrow} P'}{\text{new } a\ P \stackrel{\alpha}{\rightarrow} \text{new } a\ P'} \text{ if } \alpha \notin \{a, a'\}$$

IDENT$_t$ $$\frac{\{\vec{b}/\vec{a}\}P_A \stackrel{\alpha}{\rightarrow} P'}{A\langle\vec{b}\rangle \stackrel{\alpha}{\rightarrow} P'} \text{ if } A(\vec{a}) := P_A$$

The external interactions with other processes.

Relationships

- Structural Congruence Respects Transition: If $P \xrightarrow{\alpha} P'$ and $P \equiv Q$, then there exists some Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \equiv Q'$.
  - Structurally congruent process expressions have the same transitions.
- Reaction Agrees with τ-Transition: $P \rightarrow P'$ if and only if there exists some P'' such that $P \xrightarrow{\tau} P''$ and $P'' \equiv P'$.
  - $\rightarrow$ corresponds to the silent transition $\xrightarrow{\tau}$ (modulo congruence).

Theory of strong bisimilarity/equivalence and weak bisimilarity/observation equivalence as already discussed.

What is Mobility?

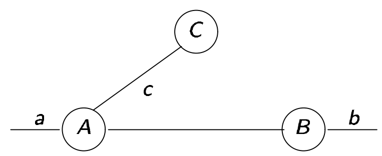
- What entities do move in what space?
  1. *Processes* move in the physical space of *computing sites*.
  2. *Processes* move in the virtual space of *linked processes*.
  3. *Links* move in the virtual space of *linked processes*.
  4. ...
- The π-Calculus is based on option (3).
  - The location of a process in a virtual space of processes is determined by its links to other processes.
    - The neighbors of a process are those processes that it can talk to.
    - Movement of a process can be described by the movement of links.
    - Option (2) can be thus reduced to option (3).
- Other calculi address option (1) more directly.
  - Ambient Calculus (Cardelli and Gordon, 1998): processes move between *ambients* (locations of activities).

The π-calculus describes a logical (not physical) view of mobility.

Mobility in CCS

$S := \text{new } c\ (A|C) \mid B$

- A and C share an internal port c.
- A and B communicate with the external world via ports a and b.

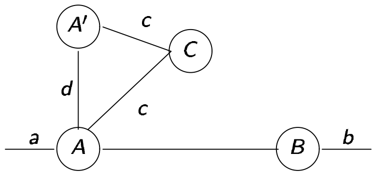


How may the shape of S change by process transitions?

Mobility in CCS

$A := a.\text{new } d\ (A|A') + c.A''$

- A may interact with environment at a.
- A then splits into A' and A'' sharing an internal port d.
  - A receives a service request at a and generates a deputy A' to which this task is delegated (e.g. a multi-threaded web server).

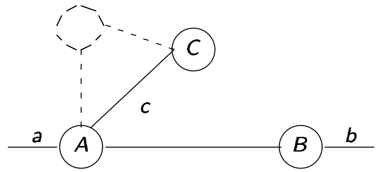


A component may generate new components.

Mobility in CCS

$A' := c.0$

- A' and C may communicate via c.
- A' then dies.
  - A' has performed the assigned task.

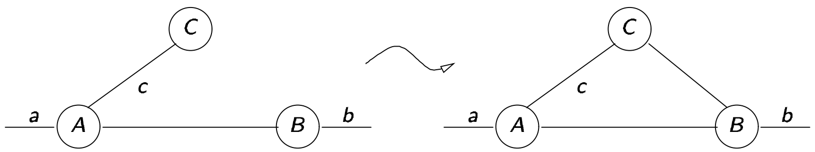


A component may disappear.

Limitations of CCS

$S := \text{new } c\ (A|C) \mid B$

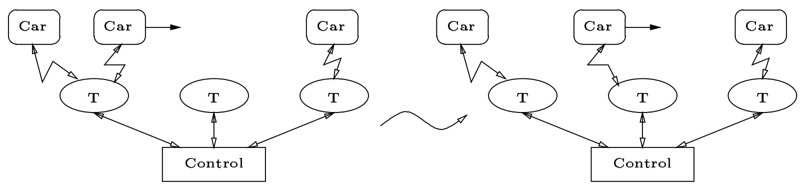
- How to achieve the following transition?



It is not possible to create new links between existing components.

An Example of Mobility

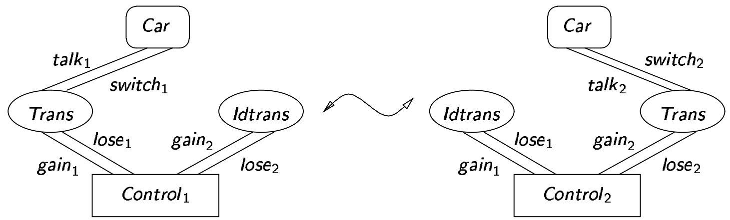
- Moving cars connected by wireless links to transmitters.
- Transmitters connected by fixed wires to a central control.
- Wireless connection of a car may be handed over from one transmitter to another.
  - Signal to original transmitter has faded by movement of car.



Virtual movement of links triggered by physical movement of cars.

A π-Calculus Model

System with one car and two transmitters.



$System :=$

$\quad \text{new } talk_1, switch_1, gain_1, lose_1, talk_2, switch_2, gain_2, lose_2$

$\quad\quad (Car\langle talk_1, switch_1\rangle | Trans\langle talk_1, switch_1, gain_1, lose_1\rangle |$

$\quad\quad Idtrans\langle gain_2, lose_2\rangle | Control_1).$

Descriptions of car and transmitters parameterized over current links.

A π-Calculus Model (Contd)

$Car(talk, switch) := \overline{talk}.Car\langle talk, switch\rangle + switch(t, s).Car\langle t, s\rangle.$

$Trans(talk, switch, gain, lose) :=$

$\quad talk.Trans\langle talk, switch, gain, lose\rangle +$

$\quad lose(t, s).\overline{switch}\langle t, s\rangle.Idtrans\langle gain, lose\rangle.$

$Idtrans(gain, lose) := gain(t, s).Trans\langle t, s, gain, lose\rangle.$

$Control_1 := \overline{lose_1}\langle talk_2, switch_2\rangle.\overline{gain_2}\langle talk_2, switch_2\rangle.Control_2.$

$Control_2 := \overline{lose_2}\langle talk_1, switch_1\rangle.\overline{gain_1}\langle talk_1, switch_1\rangle.Control_1.$

Link names may be transmitted as messages; received link names may be used for sending messages.

1. CCS Revisited

2. From CCS to the π-Calculus

3. **The π-Calculus**

The π-Calculus

- Names: $\{x, y, z, \ldots\}$.
- Action Prefixes: $\pi ::= x(y) \mid \overline{x}\langle y\rangle \mid \tau$.
  - $x(y)$ ... receive y along x.
  - $\overline{x}\langle y\rangle$ ... send y along x.
  - τ ... unobservable action.
- π-Calculus Process Expressions:

$$P ::= \sum_{i \in I} \pi_i.P_i \mid P_1|P_2 \mid \text{new } a\ P \mid !P$$

  - Summation $\sum_{i \in I} \alpha_i.P_i$ with finite indexing set I.
  - Restriction new y and input action $x(y)$ both bind name y.
  - Replication $!P$ instead of process identifiers and defining equations.

Monadic version of calculus (each message contains exactly one name).

Illustrating Reactions

$P := \text{new } z\ ((\bar{x}\langle y\rangle + z(w).\bar{w}\langle y\rangle) \mid x(u).\bar{u}\langle v\rangle \mid \bar{x}\langle z\rangle)$.

- Two possible reactions $P \rightarrow P_1$ and $P \rightarrow P_2$

  $P_1 = \text{new } z\ (0 \mid \bar{y}\langle v\rangle \mid \bar{x}\langle z\rangle)$.
  $P_2 = \text{new } z\ ((\bar{x}\langle y\rangle + z(w).\bar{w}\langle y\rangle) \mid \bar{z}\langle v\rangle \mid 0)$.
- One possible reaction $P_2 \rightarrow P_3$

  $P_3 = \text{new } z\ (\bar{v}\langle y\rangle \mid 0 \mid 0)$.

No other reactions are possible.

Structural Congruence

- Process Congruence: an equivalence relation $\simeq$ on π-calculus process expressions is a *process congruence*, if $P \simeq Q$ implies
  - $\pi.P + M \simeq \pi.Q + M$
  - $\text{new } x\ P \simeq \text{new } x\ Q$
  - $P|R \simeq Q|R,\ R|P \simeq R|Q$
  - $!P \simeq\ !Q$
- Structural Congruence: the structural congruence $\equiv$ is the process congruence defined by the following equations:
  1. Change of bound names (alpha-conversion).
  2. Reordering of terms in a summation.
  3. $P|0 \equiv P,\ P|Q \equiv Q|P,\ P|(Q|R) \equiv (P|Q)|R$.
  4. $\text{new } x\ (P|Q) \equiv P|\text{new } x\ Q$, if x not free in P.
     $\text{new } x\ 0 \equiv 0, \text{new } x\ y\ P \equiv \text{new } y\ x\ P$.
  5. $!P \equiv P \mid !P$

Alpha conversions can also occur for names bound by an input action; the replication operator can generate arbitrarily many instances of a process.

Standard Forms

- Standard Form: a process expression

  $\text{new } \vec{a}\ (M_1 \mid \ldots \mid M_m \mid !Q_1 \mid \ldots \mid !Q_n)$
  - Each M_i is a non-empty sum, each Q_n is in standard form.
  - If $m = n = 0$, the standard form is $\text{new } \vec{a}\ 0$.
  - If $\vec{a}$ is empty, the standard form is $M_1 \mid \ldots \mid M_m \mid !Q_1 \mid \ldots \mid !Q_n$.
- Theorem: Every process is structurally congruent to a standard form.

Reactions

- Reaction Relation $\rightarrow$: set of those transitions that can be inferred from the following rules:

  TAU $\tau.P + M \rightarrow P$

  REACT $(x(y).P + M)|(\bar{x}\langle z\rangle.Q + N) \rightarrow \{z/y\}P|Q$

  PAR $\dfrac{P \rightarrow P'}{P|Q \rightarrow P'|Q}$

  RES $\dfrac{P \rightarrow P'}{\text{new } x\ P \rightarrow \text{new } x\ P'}$

  STRUCT $\dfrac{P \rightarrow P'}{Q \rightarrow Q'}$, if $P \equiv Q$ and $P' \equiv Q'$

The internal reactions within a process (the external interactions will be formalized later).

The Polyadic π-Calculus

- Allow action prefixes with multiple messages.
  $x(y_1 \dots y_n).P$ and $\bar{x}\langle z_1, \dots, z_n \rangle.Q$
- Obvious encoding in monadic π-calculus:
  $x(y_1).\cdots.x(y_n).P$ and $\bar{x}\langle z_1 \rangle.\cdots.\bar{x}\langle z_n \rangle.Q$
  - Obvious encoding is wrong:
    - $x(y1, y2).P \mid \bar{x}\langle z_1, z_2 \rangle.0 \mid \bar{x}\langle z_1', z_2' \rangle.0$ should only have transitions to $\{z_1/y_1, z_2/y_2\}P$ and $\{z_1'/y_1, z_2'/y_2\}P$
    - $x(y_1).x(y2).P \mid \bar{x}\langle z_1 \rangle.\bar{x}\langle z_2 \rangle.0 \mid \bar{x}\langle z_1' \rangle.\bar{x}\langle z_2' \rangle.0$ also has transitions to $\{z_1/y_1, z_1'/y_2\}P$ and $\{z_1'/y_1, z_1/y_2\}P$.
- Correct encoding in monadic π-calculus:
  $x(w).w(y_1).\cdots.w(y_n).P$ and new w $(\bar{x}\langle w \rangle.\bar{w}\langle z_1 \rangle.\cdots.\bar{w}\langle z_n \rangle.Q)$
  - Interference on channel x is avoided by sending a fresh name w along x and then sending the components z_i one by one along w.

We can use the the polyadic π-calculus in applications but use the monadic π-calculus as the formal basis.

Recursive Definitions

- Use recursively defined process identifiers.
  Recursive definition $A(\vec{x}) := Q_A$ whose scope is process
  $P = \dots A\langle \vec{y} \rangle \dots A\langle \vec{z} \rangle \dots$
- Translated using replication as follows:
  - Invent a new name, say a, to stand for A.
  - Translate every process R to a process $\widehat{R}$ by replacing every call $A\langle \vec{w} \rangle$ by the output action $\bar{a}\langle \vec{w} \rangle$.
  - Replace the definition of A and P by
    new a $(\widehat{P} \mid !a(\vec{x}).\widehat{Q_A})$
  - Can be easily generalized to multiple recursive definitions.
- Example: $S(x) := \bar{c}(x).S(x)$ and $R := c(x).R$ in $S(y)|R$
  - new s r $(\bar{s}\langle y \rangle | \bar{r} \mid !s(x).\bar{c}\langle x \rangle.\bar{s}\langle x \rangle \mid !r.c(x).\bar{r})$

We can also use recursive process definitions in applications.