

Modeling Concurrent and Probabilistic Systems

Lecture 12: The Monadic π -Calculus

Joost-Pieter Katoen Thomas Noll

Software Modeling and Verification Group
RWTH Aachen University
`noll@cs.rwth-aachen.de`

<http://www-i2.informatik.rwth-aachen.de/i2/mcps07/>

Winter Semester 2007/08

- 1 Repetition: Modeling Mobile Concurrent Systems
- 2 Another Example: Mobile Clients
- 3 The Monadic π -Calculus

Observation: CCS imposes a **static communication structure**: if $P, Q \in \text{Proc}$ want to communicate, then both must syntactically refer to the same action name

\Rightarrow every potential communication partner known beforehand,
no dynamic passing of communication links

\Rightarrow **no mobility**

Goal: develop calculus in the spirit of CCS which supports mobility

\Rightarrow **π -calculus**

Example (Dynamic access to resources)

- Server S controls access to printer P
- Client C wishes to use P
- In **CCS**: P and C must share some action name a
 $\implies C$ could access P without being granted it by S
- In **π -calculus** :
 - initially only S has access to P (using link a)
 - using another link b , C can request access to P
- Formally:

$$\begin{array}{l}
 \underbrace{\bar{b}\langle a \rangle . S'}_S \parallel \underbrace{b(c) . \bar{c}\langle d \rangle . C'}_C \parallel \underbrace{a(e) . P}_P \\
 \xrightarrow{\tau} S' \parallel \bar{a}\langle d \rangle . C' \parallel a(e) . P' \\
 \xrightarrow{\tau} S' \parallel C' \parallel P'[e \mapsto d]
 \end{array}$$

- a : link to P
- b : link between S and C
- c : “placeholder” for a
- d : data to be printed
- e : “placeholder” for d

Example (Dynamic access to resources; continued)

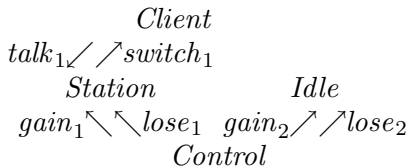
- Different rôles of action name a :
 - in interaction between S and C :
object transferred from S to C
 - in interaction between C and P :
name of communication link
- Intuitively, names represent access rights:
 - a : for P
 - b : for S
 - d : for data to be printed
- If a is only way to access P
 $\implies P$ “moves” from S to C

- 1 Repetition: Modeling Mobile Concurrent Systems
- 2 Another Example: Mobile Clients
- 3 The Monadic π -Calculus

Scenario:

- **client devices** moving around (phones, PCs, sensors, ...)
- each radioconnected to some **base station**
- stations wired to **central control**
- some event (e.g., signal fading) may cause a client to be **switched** to another station
- essential: specification of switching process (“**hand-over protocol**”)

Simplest case: two stations, one client



- Every station is in one of two **modes**: *Station* (active; four links) or *Idle* (inactive; two links)
- *Client* can **talk** via *Station*, and at any time *Control* can request *Station/Idle* to **lose/gain** *Client*:

$$\begin{aligned} \text{Station}(\text{talk}, \text{switch}, \text{gain}, \text{lose}) &= \text{talk}.\text{Station}(\text{talk}, \text{switch}, \text{gain}, \text{lose}) + \\ &\quad \text{lose}(t, s).\overline{\text{switch}}\langle t, s \rangle.\text{Idle}(\text{gain}, \text{lose}) \\ \text{Idle}(\text{gain}, \text{lose}) &= \text{gain}(t, s).\text{Station}(t, s, \text{gain}, \text{lose}) \end{aligned}$$

- If *Control* decides *Station* to lose *Client*, it issues a **new pair of channels** to be shared by *Client* and *Idle*:

$$\begin{aligned} \text{Control}_1 &= \overline{\text{lose}_1}\langle \text{talk}_2, \text{switch}_2 \rangle.\overline{\text{gain}_2}\langle \text{talk}_2, \text{switch}_2 \rangle.\text{Control}_2 \\ \text{Control}_2 &= \overline{\text{lose}_2}\langle \text{talk}_1, \text{switch}_1 \rangle.\overline{\text{gain}_1}\langle \text{talk}_1, \text{switch}_1 \rangle.\text{Control}_1 \end{aligned}$$

- *Client* can either **talk** or, if requested, **switch** to a new pair of channels:

$$\text{Client}(\text{talk}, \text{switch}) = \overline{\text{talk}}.\text{Client}(\text{talk}, \text{switch}) + \text{switch}(t, s).\text{Client}(t, s)$$

- As usual, the whole system is a **restricted composition** of processes:

$$System_1 = \text{new } L (Client_1 \parallel Station_1 \parallel Idle_2 \parallel Control_1)$$

where

$$\begin{aligned} Client_i &:= Client(talk_i, switch_i) \\ Station_i &:= Station(talk_i, switch_i, gain_i, lose_i) \\ Idle_i &:= Idle(gain_i, lose_i) \\ L &:= (talk_i, switch_i, gain_i, lose_i \mid i \in \{1, 2\}) \end{aligned}$$

- After having formally defined the π -calculus we will see that this protocol is **correct**, i.e., that the hand-over does indeed occur:

$$System_1 \longrightarrow^* System_2$$

where

$$System_2 = \text{new } L (Client_2 \parallel Idle_1 \parallel Station_2 \parallel Control_2)$$

- 1 Repetition: Modeling Mobile Concurrent Systems
- 2 Another Example: Mobile Clients
- 3 The Monadic π -Calculus

Literature on π -calculus:

- Initial research paper:
R. Milner, J. Parrow, D. Walker: *A calculus of mobile processes*, Part I/II. Journal of Inf. & Comp., 100:1–77, 1992
- Overview article:
J. Parrow: *An introduction to the π -calculus*, Chapter 8 of *Handbook of Process Algebra*, 479–543, Elsevier, 2001
- Textbook:
R. Milner: *Communicating and Mobile Systems: the π -Calculus*. Cambridge University Press, 1999

To simplify the presentation (as in Milner's book):

- 1 Monadic π -calculus with replication
(message = one name, no process identifiers)
- 2 Extension to polyadic calculus
- 3 Extension by process equations

Syntax of the Monadic π -Calculus

Definition 12.1 (Syntax of monadic π -calculus)

- Let $N = \{a, b, c, \dots, x, y, z, \dots\}$ be a set of **names**.
- The set of **action prefixes** is given by

$$\begin{array}{ll} \pi ::= x(y) & \text{(receive } y \text{ along } x) \\ \quad | \bar{x}\langle y \rangle & \text{(send } y \text{ along } x) \\ \quad | \tau & \text{(unobservable action)} \end{array}$$

- The set P^π of **π -calculus process expressions** is defined by the following syntax:

$$\begin{array}{ll} P ::= \sum_{i \in I} \pi_i.P_i & \text{(guarded sum)} \\ \quad | P_1 \parallel P_2 & \text{(parallel composition)} \\ \quad | \text{new } x P & \text{(restriction)} \\ \quad | !P & \text{(replication)} \end{array}$$

(where I finite, $x \in N$)

Conventions:

$\text{nil} := \sum_{i \in \emptyset} \pi_i.P_i$, $\text{new } x_1, \dots, x_n P := \text{new } x_1 (\dots \text{new } x_n P)$

Definition 12.2 (Free and bound names)

- The input prefix $x(y)$ and the restriction $\text{new } y P$ both **bind** y .
- Every other occurrence of a name (i.e., x in $x(y)$ and x, y in $\bar{x}\langle y \rangle$) is **free**.
- The set of bound/free names of a process expressions $P \in P^\pi$ is denoted by $bn(P)/fn(P)$, resp.

Remark: $bn(P) \cap fn(P) \neq \emptyset$ is possible

Goal: simplify definition of operational semantics by ignoring “purely syntactic” differences between processes

Definition 12.3 (Structural congruence)

$P, Q \in P^\pi$ are **structurally congruent**, written $P \equiv Q$, if one can be transformed into the other by applying the following operations and equations:

- ❶ renaming of bound names (α -conversion)
- ❷ reordering of terms in a summation (commutativity of $+$)
- ❸ $P \parallel Q \equiv Q \parallel P$, $P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R$, $P \parallel \text{nil} \equiv P$
(Abelian monoid laws for \parallel)
- ❹ $\text{new } x \text{ nil} \equiv \text{nil}$, $\text{new } x, y P \equiv \text{new } y, x P$,
 $P \parallel \text{new } x Q \equiv \text{new } x (P \parallel Q)$ if $x \notin \text{fn}(P)$ (scope extension)
- ❺ $!P \equiv P \parallel !P$ (unfolding)

Corollary 12.4 (Structural congruence)

\equiv is a congruence relation on P^π , i.e., if $P \equiv Q$ then

- ① $\pi.P + R \equiv \pi.Q + R$
- ② $P \parallel R \equiv Q \parallel R$ and $R \parallel P \equiv R \parallel Q$
- ③ $\text{new } x P \equiv \text{new } x Q$
- ④ $!P \equiv !Q$

Proof.

apply operations and equations for \equiv within respective contexts □

Theorem 12.5 (Standard form)

Every process expression is structurally congruent to a process of the standard form

$$\text{new } x_1, \dots, x_k (P_1 \parallel \dots \parallel P_m \parallel !Q_1 \parallel \dots \parallel !Q_n)$$

*where each P_i is a non-empty sum, and each Q_j is in standard form.
(If $m = n = 0$: nil; if $k = 0$: no restriction)*

Proof.

by induction on the structure of $R \in P^\pi$ (on the board)



The Reaction Relation

Thanks to Theorem 12.5, only processes in standard form need to be considered for defining the operational semantics:

Definition 12.6

The **reaction relation** $\longrightarrow \subseteq P^\pi \times P^\pi$ is generated by the rules:

$$(\text{Tau}) \frac{}{\tau.P + Q \longrightarrow P}$$

$$(\text{React}) \frac{}{(x(y).P + Q) \parallel (\bar{x}\langle z \rangle.R + S) \longrightarrow P[y \mapsto z] \parallel R}$$

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

$$(\text{Res}) \frac{P \longrightarrow P'}{\text{new } x \, P \longrightarrow \text{new } x \, P'}$$

$$(\text{Struct}) \frac{P \longrightarrow P'}{Q \longrightarrow Q'} \quad \text{if } P \equiv Q \text{ and } P' \equiv Q'$$

($P[y \mapsto z]$ replaces every free occurrence of y in P by z .)

In (React), the pair $(x(y), \bar{x}\langle z \rangle)$ is called a **redex**.)