Modeling Concurrent and Probabilistic Systems

Lecture 14: Bisimulation in the 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

Outline

Repetition: Syntax of the Monadic π-Calculus

Definition (Syntax of monadic π-calculus)

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

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

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

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

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

Conventions:

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

Repetition: The Reaction Relation

Definition

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

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

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

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

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

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

($P[y \mapsto z]$ replaces every free occurrence of y in P by z.
In (React), the pair $(x(y), \overline{x}\langle z\rangle)$ is called a redex.)

Repetition: Encoding Recursive Process Calls I

- **So far:** process replication $!P$
- **Now:** parametric process definitions of the form

$$A(x_1, \ldots, x_n) = P_A$$

  where A is a process identifier and P_A a process expression containing calls of A (and other parametric processes)
- Again: possible to simulate in basic calculus by using
  - message passing to model parameter passing to A
  - replication to model the multiple activations of A
  - restriction to model the scope of the definition of A

Repetition: Encoding Recursive Process Calls II

The encoding

- of a process definition $A(\vec{x}) = P_A$
- with right-hand side $P_A = \ldots A(\vec{u}) \ldots A(\vec{v}) \ldots$
- for main process $Q = \ldots A(\vec{y}) \ldots A(\vec{z}) \ldots$

is defined as follows:

1. Let $a \in N$ be a new name (standing for A).
2. For any process R, let $\hat{R}$ be the result of replacing every call $A(\vec{w})$ by $\overline{a}\langle\vec{w}\rangle$.
3. Replace Q by $Q' := \mathsf{new}\, a\, (\hat{Q} \parallel !a(\vec{x}).\hat{P}_A)$.

(In the presence of more than one process identifier, Q' will contain a replicated component for each definition.)

Encoding Recursive Process Calls II

Example

- One-place buffer: $B(in, out) = in(x).\overline{out}\langle x\rangle.B(in, out)$
- Main process: $Q := \overline{in}\langle y\rangle \parallel B(in, out) \parallel out(z)$
- Encoding:

$$Q' := \mathsf{new}\, b\, (\overline{in}\langle y\rangle \parallel \overline{b}\langle in, out\rangle \parallel out(z) \parallel \underbrace{!b(in, out).in(x).\overline{out}\langle x\rangle.\overline{b}\langle in, out\rangle}_{=:P})$$

$$\equiv$$

$$\mathsf{new}\, b\, (\overline{in}\langle y\rangle \parallel \overline{b}\langle in, out\rangle \parallel out(z) \parallel b(in, out).in(x).\overline{out}\langle x\rangle.\overline{b}\langle in, out\rangle \parallel P)$$

$$\downarrow$$

$$\mathsf{new}\, b\, (\overline{in}\langle y\rangle \parallel out(z) \parallel in(x).\overline{out}\langle x\rangle.\overline{b}\langle in, out\rangle \parallel P)$$

$$\downarrow$$

$$\mathsf{new}\, b\, (out(z) \parallel \overline{out}\langle y\rangle.\overline{b}\langle in, out\rangle \parallel P)$$

$$\downarrow$$

$$\mathsf{new}\, b\, (\overline{b}\langle in, out\rangle \parallel P)$$

$$\downarrow$$

$$\mathsf{new}\, b\, (in(x).\overline{out}\langle x\rangle.\overline{b}\langle in, out\rangle \parallel P)$$

Outline

Commitments

- **Goal:** establish equivalence relations between π-calculus processes (e.g., for establishing the correctness of encodings)
- **But:** reaction relation $\longrightarrow \subseteq P^\pi \times P^\pi$ is too coarse
- For example, $x(y).\mathsf{nil}$ and $\overline{x}\langle z\rangle.\mathsf{nil}$ both have no reactions
  $\Longrightarrow$ bisimilar w.r.t. $\longrightarrow$, but different reaction capabilities
- **Solution:** introduce commitments in π-calculus, corresponding to transitions in CCS
  - CCS: $P \parallel Q = (\ a.P' + \ldots) \parallel (\ \overline{a}.Q' + \ldots) \xrightarrow{\tau} P' \parallel Q'$

    $\downarrow a$ P' ; $\downarrow \overline{a}$ Q' — commitments
  - π-calculus:

    $$P \parallel Q = (\ x(\vec{y}).P' + \ldots) \parallel (\ \overline{x}\langle\vec{z}\rangle.Q' + \ldots) \longrightarrow P'[\vec{y} \mapsto \vec{z}] \parallel Q'$$

    $\downarrow x$ $(\vec{y}).P'$ — abstraction ; $\downarrow \overline{x}$ $\langle\vec{z}\rangle.Q'$ — concretion ; commitments

Abstraction and Concretions

Definition 14.1

- An abstraction of arity $n \in \mathbb{N}$ is of the form $(\vec{x}).P$, where $\vec{x} = (x_1, \ldots, x_n)$ and $P \in P^\pi$.
- A concretion of arity $n \in \mathbb{N}$ is of the form $\mathsf{new}\,\vec{x}\,\langle\vec{y}\rangle.P$, where $\vec{y} = (y_1, \ldots, y_n)$, $\vec{x} \subseteq \vec{y}$, and $P \in P^\pi$.
- An agent is an abstraction or a concretion (notation: A^π).

Remarks:

- We use
  - F, G to denote abstractions,
  - C, D to denote conretions,
  - $A, B \in A^\pi$ to denote agents
- Note: a process $P \in P^\pi$ is both an abstraction and a concretion of arity 0
- $\equiv / fn / bn$ also extends to agents
- Guarded sum now considered as $\sum \alpha_i A_i$ where $\alpha_i \in N \cup \overline{N} \cup \{\tau\}$

Application

Definition 14.2

The application $F \mathbin{@} C$ (where F and C are of equal arity) is defined as follows, assuming $\vec{z} \cap fn((\vec{x}).P) = \emptyset$:

$$(\vec{x}).P \mathbin{@} \textsf{new}\, \vec{z}\, \langle\vec{y}\rangle.Q := \textsf{new}\, \vec{z}\, (P[\vec{x} \mapsto \vec{y}] \parallel Q)$$

Remarks:

- The (React) rule can now be represented as

$$\text{(React)}\,\frac{}{(xF + P) \parallel (\overline{x}C + Q) \longrightarrow F \mathbin{@} C}$$

- We add the following equations for structural congruence (assuming $z \notin \vec{x}$ and $\vec{x} \cap fn(Q) = \emptyset$):

$$\textsf{new}\, z\, ((\vec{x}).P) \equiv (\vec{x}).\textsf{new}\, z\, P$$
$$((\vec{x}).P) \parallel Q \equiv (\vec{x}).(P \parallel Q)$$

Commitment Relation

Definition 14.3

The commitment relation $\longrightarrow \subseteq A^\pi \times (N \cup \overline{N} \cup \{\tau\}) \times A^\pi$ is generated by the following rules:

$$\text{(SUM)}\ \frac{}{\alpha.A + M \xrightarrow{\alpha} A} \qquad \text{(REACT)}\ \frac{P \xrightarrow{x} F \quad Q \xrightarrow{\overline{x}} C}{P \parallel Q \xrightarrow{\tau} F \mathbin{@} C}$$

$$\text{(PAR)}\ \frac{P \xrightarrow{\alpha} A}{P \parallel Q \xrightarrow{\alpha} A \parallel Q} \qquad \text{(RES)}\ \frac{P \xrightarrow{\alpha} A \quad \alpha \notin \{x, \bar{x}\}}{\mathsf{new}\, x\, P \xrightarrow{\alpha} \mathsf{new}\, x\, A}$$

$$\text{(STRUCT)}\ \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q}{P \xrightarrow{\alpha} Q}$$

Example 14.4

$y(z).P \parallel \mathsf{new}\, x\, (\overline{y}\langle x\rangle.Q + \overline{w}\langle v\rangle.R) \xrightarrow{\tau} \mathsf{new}\, x\, (P[z \mapsto x] \parallel Q)$
(if $x \notin fn(P)$; on the board)

Outline

Strong Bisimulation I

Problem: the target $A \in A^\pi$ of a commitment $P \xrightarrow{\alpha} A$ (where $\alpha \neq \tau$) is not necessarily a process, but an abstraction or a concretion

$\implies$ extend process equivalences to agent equivalences

Definition 14.5

Let $\rho \subseteq P^\pi \times P^\pi$ be a binary relation on processes. The extension of ρ to agents, $\rho \subseteq A^\pi \times A^\pi$, is defined by

$$\begin{aligned} F \rho G &:\iff \text{for all } \vec{y} : (F \mathbin{@} \langle \vec{y} \rangle.\mathsf{nil}) \rho (G \mathbin{@} \langle \vec{y} \rangle.\mathsf{nil}) \\ C \rho D &:\iff \text{there exist } P \rho Q \text{ such that } C \equiv \mathsf{new}\, \vec{z}\, \langle \vec{y} \rangle.P \text{ and} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\quad D \equiv \mathsf{new}\, \vec{z}\, \langle \vec{y} \rangle.Q \end{aligned}$$

Strong Bisimulation II

Definition 14.6 (Strong bisimulation)

A relation $\rho \subseteq P^\pi \times P^\pi$ is called a strong bisimulation if $P \rho Q$ implies

1. $P \xrightarrow{\alpha} A \implies$ ex. $B \in A^\pi$ such that $Q \xrightarrow{\alpha} B$ and $A \rho B$
2. $Q \xrightarrow{\alpha} B \implies$ ex. $A \in A^\pi$ such that $P \xrightarrow{\alpha} A$ and $A \rho B$

Two agents $A, B \in A^\pi$ are called strongly bisimilar (notation: $A \dot{\sim} B$) if $A \rho B$ for some strong bisimulation ρ.

Lemma 14.7

1. *$\dot{\sim}$ is a strong bisimulation (the largest one).*
2. *$\dot{\sim}$ is an equivalence relation.*

Proof.

similar to CCS case (Theorem 4.2) □

Example 14.8

on the board

Strong Bisimulation as a Congruence I

- **Problem:** strong bisimulation is not a π-calculus process congruence
- Not preserved by input prefix
- **Example:** $P := \overline{x} \parallel y$, $Q := \overline{x}.y + y.\overline{x}$
  - $P \dot{\sim} Q$ (obvious) but
  - $P[y \mapsto x] \not\dot{\sim} Q[y \mapsto x]$
    (since $P[y \mapsto x] = \overline{x} \parallel x \xrightarrow{\tau} \mathsf{nil}$ and $Q[y \mapsto x] = \overline{x}.x + x.\overline{x} \not\xrightarrow{\tau}$)

$\Longrightarrow$ $(y).P \;@\; \langle x \rangle.\mathsf{nil} \not\dot{\sim} (y).Q \;@\; \langle x \rangle.\mathsf{nil}$ (Def. 14.2 of @)

$\Longrightarrow$ $(y).P \not\dot{\sim} (y).Q$ (Def. 14.5 of $\dot{\sim}$ for abstractions)

Strong Bisimulation as a Congruence II

Definition 14.9 (Strong congruence)

Two processes $P, Q \in P^{\pi}$ are called strongly congruent ($P \sim Q$) if $P\sigma \dot{\sim} Q\sigma$ for every substitution $\sigma : N \to N$.

Lemma 14.10

$\sim$ is the largest congruence in $\dot{\sim}$.

Proof.

omitted □