Distributed Systems: the Pi-calculus

Model-based Design of Distributed and Mobile Systems Lecture 9: Spring 2005

Mikkel Bundgaard
mikkelbu -at- itu.dk

Department of Theoretical Computer Science
IT University of Copenhagen
Overview of the Lecture

The structure of the lecture:

- Selected Exercises
- Behavioural Equivalences
  - Trace Equivalence
  - Bisimulation Equivalence
  - Examples and MWB
  - Equivalence and Congruence Relation
- The Spi Calculus
  - Syntax
  - Structural Equivalence and Reaction relation
  - Examples and Properties
- Exercises
Selected Exercises

Show: \( \bar{x}\langle y \rangle .0 | (\nu y) x(z) . Q\langle y, z \rangle \rightarrow (\nu y') Q\langle y', y \rangle \)

\( \bar{x}\langle y \rangle .0 | (\nu y) x(z) . Q\langle y, z \rangle \equiv \)

\( (\nu y')( (x(z) . Q\langle y', z \rangle + 0) | (\bar{x}\langle y \rangle . 0 + 0) ) \rightarrow \)

\( (\nu y')( Q\langle y', y \rangle | 0) \equiv (\nu y') Q\langle y', y \rangle \)

Apply the translation to \( x(y_1, y_2) . P | \bar{x}\langle z_1, z_2 \rangle . Q | \bar{x}\langle z'_1, z'_2 \rangle . Q' \)

\( x(w) . w(y_1) . w(y_2) . P | (\nu w) (\bar{x}\langle w \rangle . \bar{w}\langle z_1 \rangle . \bar{w}\langle z_2 \rangle . Q) | (\nu w) (\bar{x}\langle w \rangle . \bar{w}\langle z'_1 \rangle . \bar{w}\langle z'_2 \rangle . Q') \)

Evaluate the expressions

- \( \text{True}_a | \text{Case}_a(P, Q) \rightarrow P \)
- \( \text{False}_a | \text{Case}_a(P, Q) \rightarrow Q \)
Selected Exercises

How would you implement logical and, logical or, and logical not?
(Using Case)

\[ Not_{b,r} = Case_{b}(False_r, True_r) \]

\[ And_{b_1,b_2,r} = Case_{b_1}(Case_{b_2}(True_r, False_r), False_r) \]
Behavioural Equivalences

- Often systems are informally specified using “behave like” statements
- But how can we formalise behavioural equivalence
- We would like that the equivalence satisfy the following requirements
  1. It should be a reflexive, symmetric, and transitive relation
  2. Processes that may terminate (deadlock) should not be equivalent to processes that may not terminate (deadlock)
  3. It should be a congruence. That is, if a component $Q$ of $P$ is replaced by an equivalent component $Q'$ yielding $P'$, then $P$ and $P'$ should also be equivalent
  4. Two processes should be equivalent iff they satisfy exactly the same properties expressible in a nice modal or temporal logic
  5. (it should abstract from silent actions)
Trace Equivalence (LTS and CCS)

- Note that we work in CCS, since the theory is simpler here.
- Processes are trace equivalent exactly when they have the same traces (from their initial state).
- Let $P \xrightarrow{a_1 \ldots a_n} Q$ if $P \xrightarrow{a_1} P_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} Q$ for some $P_1, \ldots, P_{n-1}$.
- Let $S(P) = \{ s \in A^* \mid \exists Q. P \xrightarrow{s} Q \}$ be all traces of $P$. Then $P \approx_S Q$ if and only if $S(P) = S(Q)$.
- $a.b.0 + a.c.0 \approx_S a.(b.0 + c.0)$, but $[a] \langle b \rangle tt$ distinguish the two (so it is not a congruence $(a, b)(\_ | \overline{a} \overline{b})$).
- Let $P = a.P$ and $Q = P + a.0$. Then $P \approx_S Q$, but $Q$ can deadlock, whereas $P$ cannot.
- Trace equivalence is inadequate for non-deterministic systems.
Bisimulation Equivalence — Game Definition

- **Board**: Transition systems of $P$ and $Q$
- **Material**: Two (identical) pebbles, initially on the states $P$ and $Q$
- **Players**: $R$ (refuter) and $V$ (verifier), $R$ and $V$ take turns, $R$ moves first.

- $R$-move: Choose any of the two pebbles
  - Take any transition

- $V$-move: Choose the other pebble
  - Take any transition having the same label as the one chosen by $R$.

- $R$ wins if: $V$ cannot reply to his (or her) last move
- $V$ wins if: $R$ cannot move or the game goes on forever.
  (i.e., a draw counts as a win for $V$).
Bisimulation Equivalence

Two processes should be considered equivalent only if they infinitely can mimic each other.

A relation $R$ is a bisimulation if $P R Q$ then

- $P \xrightarrow{a} P'$ implies $\exists Q'. Q \xrightarrow{a} Q'$ such that $P'RQ'$
- $Q \xrightarrow{a} Q'$ implies $\exists P'. P \xrightarrow{a} P'$ such that $P'RQ'$

$P$ and $Q$ are bisimilar, $P \sim Q$, if there exists a bisimulation between them.

Theorem $\sim$ implies $\approx_S$ (meaning $\sim \subseteq \approx_S$)
($\approx_S = \sim$, if LTS is deterministic)

$\approx_S$ does no imply $\sim$, why?
Examples

- \( a.b.0 + a.c.0 \not\sim a.(b.0 + c.0) \)
- \( P \not\sim Q, \) if \( P = a.P \) and \( Q = P + a.0 \)
- So, \( \sim \) captures non-determinism and deadlock, and \textbf{when} choices have been made
- \( a.b.0 + a.c.0 \sim a.c.0 + a.b.0 \) and \( a.(b.0 + c.0) \sim a.(c.0 + b.0) \) because

\[
\{ (a.b.0 + a.c.0, a.c.0 + a.b.0), \\
(a.(b.0 + c.0), a.(c.0 + b.0)), \\
(b.0 + c.0, c.0 + b.0), \\
(b.0, b.0), \\
(c.0, c.0), \\
(0, 0) \}
\]

is a \textbf{bisimulation}

- it defines a \textbf{winning strategy} for the verifier in the pebble game
The command `eqd (name_1, ..., name_n) agent1 agent2` checks whether `agent1` and `agent2` are strong bisimulation equivalent under the assumption that the names in `(name_1, ..., name_n)` are distinct. We use these names to differentiate the different free names of the term under consideration.
Examples — in MWB

MWB>agent \( P(a,b,c) = a \cdot b \cdot 0 + a \cdot c \cdot 0 \)
MWB>agent \( Q(a,b,c) = a \cdot (b \cdot 0 + c \cdot 0) \)
MWB>eqd (a,b,c) \( P\langle a,b,c \rangle \ Q\langle a,b,c \rangle \)
The two agents are NOT equal.

MWB>agent \( R(a) = a \cdot R\langle a \rangle \)
MWB>agent \( S(a) = R\langle a \rangle + a \cdot 0 \)
MWB>eqd (a) \( R\langle a \rangle \ S\langle a \rangle \)
The two agents are NOT equal.

MWB>agent \( P_1(a,b,c) = a \cdot c \cdot 0 + a \cdot b \cdot 0 \)
MWB>eqd (a,b,c) \( P\langle a,b,c \rangle \ P_1\langle a,b,c \rangle \)
The two agents are equal.
Bisimulation relation size = 4.

MWB>agent \( Q_1(a,b,c) = a \cdot (c \cdot 0 + b \cdot 0) \)
MWB>eqd (a,b,c) \( Q\langle a,b,c \rangle \ Q_1\langle a,b,c \rangle \)
The two agents are equal.
Bisimulation relation size = 3.
Examples cont.

Let $P = b.0 \mid \overline{a}.c.0$, $Q = a.b.0 \mid c.0$, and let $R = a.b.0 \mid \overline{a}.c.0$ then

\[
P \sim b.\overline{a}.c.0 + \overline{a}.(b.c.0 + c.b.0)
\]

\[
Q \sim a.(b.c.0 + c.b.0) + c.a.b.0
\]

\[
R \sim a.P + \overline{a}.Q + \tau.(b.c.0 + c.b.0)
\]

because there exists a bisimulation

\{
(P, b.\overline{a}.c.0 + \overline{a}.(b.c.0 + c.b.0)),
(R, a.P + \overline{a}.Q + \tau.(b.c.0 + c.b.0)),
(Q, a.(b.c.0 + c.b.0) + c.a.b.0),
(b.0 \mid c.0, b.c.0 + c.b.0),
(S, S), ((0 \mid S), S), (S, (0 \mid S)) \mid S \in \mathcal{P}
\}
Examples cont. — in MWB

MWB>agent P(a,b,c) = b.0 \mid 'a.c.0
MWB>agent Q(a,b,c) = a.b.0 \mid c.0
MWB>agent R(a,b,c) = a.b.0 \mid 'a.c.0

MWB>eqd (a,b,c) P(a,b,c) = b.'a.c.0 + 'a.(b.c.0 + c.b.0)
The two agents are equal.
Bisimulation relation size = 6.

MWB>eqd (a,b,c) Q(a,b,c) = c.a.b.0 + a.(b.c.0 + c.b.0)
The two agents are equal.
Bisimulation relation size = 6.

MWB>eqd (a,b,c) R(a,b,c) = a.P(a,b,c) + 'a.Q(a,b,c) + t.(b.c.0 + c.b.0)
The two agents are equal.
Bisimulation relation size = 10.
Bisimulation is an Equivalence Relation

Often we want equality relations, like $\sim$, to satisfy certain natural properties.

- An **equivalence relation** is a binary relation $\mathcal{R}$ such that $\mathcal{R}$ is
  - reflexive ($P \mathcal{R} P$)
  - symmetric ($P \mathcal{R} Q$ implies $Q \mathcal{R} P$)
  - transitive ($P \mathcal{R} Q$ and $Q \mathcal{R} R$ implies $P \mathcal{R} R$)

- **Theorem** $\sim$ is an equivalence relation, which allows for equational reasoning
Bisimulation is a Congruence

- We would like that equivalent processes are interchangeable inside larger systems.

- A context $C$ is a process expression with a hole $[]$,

  $$C ::= [] \mid 0 \mid \alpha.C \mid C + P \mid P + C$$

  $$\mid C \mid P \mid P \mid C \mid (a)C$$

- An equivalence relation $\mathcal{R}$ is a congruence if for all $C$

  $$P \mathcal{R} Q \text{ implies } C[P] \mathcal{R} C[Q]$$

- Remember the definition of structural congruence last time, as the smallest congruence that fulfills certain rules

- Theorem $\sim$ is a congruence.
Example

Let $P = b.0 \mid \overline{a}.c.0$, $Q = a.b.0 \mid c.0$ and let $R = a.b.0 \mid \overline{a}.c.0$ then (from before)

$$P \sim b.\overline{a}.c.0 + \overline{a}.(b.c.0 + c.b.0)$$

$$Q \sim a.(b.c.0 + c.b.0) + c.a.b.0$$

$$R \sim a.P + \overline{a}.Q + \tau.(b.c.0 + c.b.0)$$

Since $\sim$ is a congruence it then follows that $R$ is bisimilar to

$$a.(b.\overline{a}.c.0 + \overline{a}.(b.c.0 + c.b.0)) + \overline{a}.(a.(b.c.0 + c.b.0) + c.a.b.0) + \tau.(b.c.0 + c.b.0)$$

Notice, that parallel composition doesn’t occur on the right hand side.

**Theorem (Expansion)** For any $P$ there exists $Q$ with no parallel components such that $P \sim Q$.

Hence, it cannot differentiate concurrency from interleaving.
The Spi Calculus

- We augment the pi-calculus with primitives for cryptography
  - A middle ground between specification and implementation
- We use the Spi Calculus for studying authentication protocols (e.g. the explicit representation of the use of cryptography in protocols)
- We model a possible malicious attacker as an arbitrary environment
  - Hence, we need not model the attacker explicitly
- The scoping of the pi-calculus ensures that the environment cannot access restricted channels
  - Restriction and scope extrusion represents the possession and communication of secrets
The pi-Calculus — Syntax

The syntax of the pi-calculus with pairs and numbers

Terms \( L, M, N \) ::= \( n \) name
\( (M, N) \) pair
0 zero
\( \text{suc}(M) \) successor
\( x \) variable

Agent \( P, Q, R \) ::= \( 0 \) inactive
\( \overline{M} \langle N \rangle . P \) output
\( M(x). P \) input
\( P \mid Q \) parallel
\( (\nu a) P \) restriction
\( !P \) replication
\( [M \text{ is } N] P \) match
\( \text{let}(x, y) = M \text{ in } P \) pair splitting
\( \text{case } M \text{ of } 0 : P \text{ suc}(x) : Q \) integer case
The Spi-Calculus — syntax

- We augment the syntax of the previous slide
- To the syntactic category of terms we add

\[
\{M\}_N \text{ the term } M \text{ encrypted with the key } N
\]

- and to processes

\[
\text{case } L \text{ of } \{x\}_N \text{ in } P \text{ shared-key decryption}
\]

- Assumptions
  - The only way to \textit{decrypt} a an encrypted packet is to know the corresponding key
  - An encrypted packet \textit{does not reveal} the key that was used to encrypt it
  - The decryption algorithm can \textit{check} whether a ciphertext was encrypted with the expected key
The Spi-Calculus — Structural Equivalence

\[ \begin{align*}
!P & > P \; |!P \\
[M \text{ is } M]P & > P \\
let(x, y) = (M, N) \text{ in } P & > P[M/x][N/y] \\
\text{case } 0 \text{ of } 0 : P \; suc(x) : Q & > P \\
\text{case } suc(M) \text{ of } 0 : P \; suc(x) : Q & > Q[M/x] \\
\text{case } \{M\}_N \text{ of } \{x\}_N \text{ in } P & > P[M/x]
\end{align*} \]

\textbf{Structural equivalence} is defined in a different way than expected, but gives rise to the same relation (e.g. that it is an equivalence relation is explicitly stated as rules)

\[ P > Q \quad \text{Therefore,} \quad P \equiv Q \]
The Spi-Calculus — Reduction Semantics

**Reaction Rules**

\[
\begin{align*}
    m(x).Q \parallel \overline{m}(N).P & \rightarrow Q[N/x] \parallel P \\
    P \rightarrow P' & \Rightarrow P \parallel Q \rightarrow P' \parallel Q \\
    (\nu x)P & \rightarrow (\nu x)P'
\end{align*}
\]
Examples — Channel Establishment

Message 1 $A \rightarrow S : c_{AB}$ on $c_{AS}$
Message 2 $S \rightarrow B : c_{AB}$ on $c_{SB}$
Message 3 $A \rightarrow B : M$ on $c_{AB}$

For the next slide

- Assume that $F$ is a function that we can apply
- and that bound parameters of the protocol cannot occur free in $F$
Examples — Protocol and Specification

Protocol

\[ A(M) \triangleq (\nu c_{AB})c_{AS}(c_{AB})_cA_B(M) \]
\[ S \triangleq c_{AS}(x)_cS_B(x) \]
\[ B \triangleq c_{SB}(x).x(y).F(y) \]
\[ Inst(M) \triangleq (\nu c_{AS})(\nu c_{SB})(A(M) \mid S \mid B) \]

Specification

\[ A(M) \triangleq (\nu c_{AB})c_{AS}(c_{AB})_cA_B(M) \]
\[ S \triangleq c_{AS}(x)_cS_B(x) \]
\[ B_{spec} \triangleq c_{SB}(x).x(y).F(M) \]
\[ Inst_{spec}(M) \triangleq (\nu c_{AS})(\nu c_{SB})(A(M) \mid S \mid B_{spec}(M)) \]
Examples — Satisfied Properties

- **Authenticity:** $\text{Inst}(M) \simeq \text{Inst}_{\text{spec}}(M)$, for any $M$
  So $B$ always applies $F$ to the message $M$ that $A$ sends, an attacker cannot cause $B$ to apply $F$ to some other message.

- **Secrecy:** $\text{Inst}(M) \simeq \text{Inst}(M')$, if $F(M) \simeq F(M')$, for any $M, M'$
  So the message $M$ cannot be read in transit from $A$ to $B$. If $F$ does not reveal $M$ then the whole protocol does not

- These properties hold because of the scoping rules of the pi-calculus
Examples — Simple: Using Cryptography

Now we send the message on public channels (so the system can get stuck) using a shared secret key

🛠️ Message 1 $A \rightarrow B : \{M\}_{K_{AB}}$ on $c_{AB}$

**Protocol**

$$A(M) \triangleq \overline{c_{AB}}(\{M\}_{K_{AB}})$$

$$B \triangleq c_{AB}(x).\text{case } x \text{ of } \{y\}_{K_{AB}} \text{ in } F(y)$$

$$\text{Inst}(M) \triangleq (\nu K_{AB})(A(M) \mid B)$$

**Specification**

$$A(M) \triangleq \overline{c_{AB}}(\{M\}_{K_{AB}})$$

$$B_{\text{spec}} \triangleq c_{AB}(x).\text{case } x \text{ of } \{y\}_{K_{AB}} \text{ in } F(M)$$

$$\text{Inst}(M)_{\text{spec}} \triangleq (\nu K_{AB})(A(M) \mid B_{\text{spec}}(M))$$
Examples — Key Establishment

Each principal shares a key with the server

Message 1 $A \rightarrow S : \{K_{AB}\}_{K_{AS}}$ on $c_{AS}$
Message 2 $S \rightarrow B : \{K_{AB}\}_{K_{SB}}$ on $c_{SB}$
Message 3 $A \rightarrow B : \{M\}_{K_{AB}}$ on $c_{AB}$

Note that our equivalence is coarse-grained enough to equate $(\nu K)c\langle\{M\}_K\rangle$ and $(\nu K)c\langle\{M'\}_K\rangle$
Examples — Protocol and Specification

Protocol

\[ A(M) \triangleq (\nu K_{AB})(c_{AS}\langle\{K_{AB}\}_KAS\rangle).c_{AB}\langle\{M\}_KAB\rangle \]

\[ S \triangleq c_{AS}(x).case \ x \ of \ \{y\}_KAS \ in \ c_{SB}\langle\{y\}_KSB\rangle \]

\[ B \triangleq c_{SB}(x).case \ x \ of \ \{y\}_KSB \ in \ c_{AB}(z).case \ z \ of \ \{w\}_y \ in \ F(w) \]

\[ Inst(M) \triangleq (\nu K_{AS})(\nu k_{sb})(A(M) \mid S \mid B) \]

Specification

\[ A(M) \triangleq (\nu K_{AB})(c_{AS}\langle\{K_{AB}\}_KAS\rangle).c_{AB}\langle\{M\}_KAB\rangle \]

\[ S \triangleq c_{AS}(x).case \ x \ of \ \{y\}_KAS \ in \ c_{SB}\langle\{y\}_KSB\rangle \]

\[ B \triangleq c_{SB}(x).case \ x \ of \ \{y\}_KSB \ in \ c_{AB}(z).case \ z \ of \ \{w\}_y \ in \ F(M) \]

\[ Inst_{spec}(M) \triangleq (\nu K_{AS})(\nu k_{sb})(A(M) \mid S \mid B_{spec}(M)) \]
Possible 4-week project

- Since the Mobility Workbench is not a user friendly tool a possible 4-week project could be to create a new workbench.
- Depending on our experience with programming languages, I could supply the code for parsers and lexers.
- You can then program the methods for:
  - finding and performing transitions (or reactions)
  - running and visualising the process
  - ...

- For more project proposals see (updated soon)

http://www.itu.dk/research/theory/CoMo/student.html
http://www.itu.dk/research/theory/bpl/projects/projectlist.html
Exercises — Bisimulation Equivalences

- Why is trace equivalence appropriate for deterministic systems?

- Give the bisimulation of size 10 reported by MWB on slide 13.

- Show a process that does not contain parallel composition and that is bisimulation equivalent to \((a)(a.0 + (\overline{a}.b.0 | a.b.0))\).

- Define a bisimulation containing the two processes above.
Exercises — Pi-calculus Semantics

- Step through
  agent \( P(b,c,d) = (\forall a)(b^{<a>}.0 \mid a(x).x.0) \mid b(c).c^{<d>}.0 \)

in MWB, what is going on?

- Eliminate the substitution in
  \( (\forall b)(\overline{ab}.0 + b(x).\overline{xy}.0)\{b/a, b/x, b/y\} \)

- Prove
  \[
  ((\forall b)\overline{ab}.P + (\forall c)\overline{ac}.P') \mid a(b).Q \quad \rightarrow \quad (\forall b')(P\{b'/b\} \mid Q\{b'/b\})
  \]

using the structural congruence and reaction relation from last lecture
Exercises — SPI Calculus

1. Write out the malicious replay attacker from p. 7 on the article

2. (and how it can invalidate the authenticity equation)

3. Show that the new protocol (Section 3.2.4) cannot be broken by this attacker
Exercises — Do the mandatory Exercise