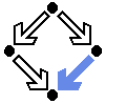
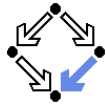


The pi-Calculus (Part 2)

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>

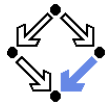


1. Strong Equivalence

2. Observation Equivalence

3. Extensions of the π -Calculus

Process Commitments

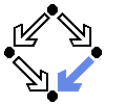


When can a reaction $P|Q \rightarrow P'|Q'$ occur?

- **CCS:** $P|Q = (\dots + a.P')|(\dots + \bar{a}.Q')$
 - Transitions $P \xrightarrow{a} P'$ and $Q \xrightarrow{\bar{a}} Q'$ are necessary.
 - $P \xrightarrow{a} P'$... "commitment" of P to take part in a reaction involving a .
 - Residue of reaction is $P'|Q'$.
- **π -Calculus:** $P|Q = (\dots + x(\vec{y}).P')|(\dots + \bar{x}\langle\vec{z}\rangle.Q')$
 - Commitments $P \xrightarrow{x} (\vec{y}).P'$ and $Q \xrightarrow{\bar{x}} \langle\vec{z}\rangle.Q'$ are necessary.
 - Abstraction $(\vec{y}).P'$, concretion $\langle\vec{z}\rangle.Q'$.
 - Residue of reaction is $(\vec{y}).P'@(\vec{z}).Q' = \{\vec{z}/\vec{y}\}P'|Q'$.
 - $F@C$ is the application of an abstraction F and a concretion C .

Revise transition rules of CCS to commitment rules of the π -calculus.

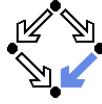
Abstractions and Concretions



- **Abstraction:** $(\vec{x}).P$
 - Bound names \vec{x} , process part P .
 - If $|\vec{x}| = n$, then the abstraction has arity n .
 - Two abstractions F and G are structurally congruent ($F \equiv G$), if they have same arity and, up to alpha-conversion of the bound names, their process parts are structurally congruent.
- **Concretion:** $\text{new } \vec{x} \langle\vec{y}\rangle.P$
 - Restricted names \vec{x} , prefix $\text{new } \vec{x} \langle\vec{y}\rangle$, process part P .
 - If $|\vec{y}| = n$, then the concretion has arity n .
 - Two concretions C and D are structurally congruent ($C \equiv D$), if they have the same arity and, up to alpha-conversion and re-ordering of the restricted names, their prefixes are identical and their process parts are structurally congruent.

Concretions may include restrictions to keep names secret between the receiver of a message and (the residue of) the sender.

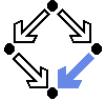
Agents



- **Agent:** an abstraction or a concretion.
 - Every process is an abstraction and a concretization of arity 0.
 - Every process is thus an agent.
- **Generalized Summation:** $\sum_{i \in I} \alpha_i A_i$
 - $\alpha_i A_i$ has form $x F$, $\bar{x} C$, or τP .
 - Output action may include a restriction.
 - Example: $\bar{x}(\text{new } y_1 \langle y_1 y_2 \rangle . P)$
- **Generalized Reaction Rule:**
REACT $(x F + M) | (\bar{x} C + N) \rightarrow F @ C$
 - Application $F @ C$ yields a process:
 $(\bar{x}) . P @ \text{new } \bar{z} \langle \bar{y} \rangle . Q := \text{new } \bar{z} \langle \{\bar{y} / \bar{x}\} P | Q$
 - $|\bar{x}| = |\bar{y}|$
 - \bar{z} not free in $(\bar{x}) . P$.

The commitment rules operate on agents.

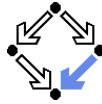
Generalized Process Operations



Agents ($\text{new } z A$) and $A | Q$ of same kind and arity as agent A .

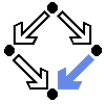
- **Generalized Restriction:**
 We assume $z \notin \bar{x}$.
 - $\text{new } z ((\bar{x}) . P) := (\bar{x}) . \text{new } z P$
 - $\text{new } z (\text{new } \bar{x} \langle \bar{y} \rangle . P) := \begin{cases} \text{new } z \bar{x} \langle \bar{y} \rangle . P & \text{if } z \in \bar{y} \\ \text{new } \bar{x} \langle \bar{y} \rangle . \text{new } z P & \text{otherwise} \end{cases}$
- **Generalized Composition:**
 We assume $z \notin \bar{x}$ and \bar{x} not free in Q .
 - $((\bar{x}) . P) | Q := (\bar{x}) . (P | Q)$
 - $(\text{new } \bar{x} \langle \bar{y} \rangle . P) | Q := \text{new } \bar{x} \langle \bar{y} \rangle . (P | Q)$
- **Structural Congruences:**
 - $A | (P | Q) \equiv (A | P) | Q$
 - $\text{new } x (A | Q) \equiv \begin{cases} A | \text{new } x Q & (x \text{ not free in } A) \\ \text{new } x A | Q & (x \text{ not free in } Q) \end{cases}$
 - $(F | P) @ (C | Q) \equiv (F @ C) | P | Q$
 - $\text{new } x (F @ C) \equiv \begin{cases} F @ \text{new } x C & (x \text{ not free in } F) \\ \text{new } x F @ C & (x \text{ not free in } C) \end{cases}$

Commitment Rules



- **Commitment Relation** $\overset{\alpha}{\rightarrow}$: Set of those commitments that can be inferred from the following rules, together with alpha-conversion (where α is either a label or τ):
 - SUM_c** $M + \alpha A + N \overset{\alpha}{\rightarrow} A$
 - L-REACT_c** $\frac{P \overset{x}{\rightarrow} F \quad Q \overset{\bar{x}}{\rightarrow} C}{P | Q \overset{\tau}{\rightarrow} F @ C}$
 - R-REACT_c** $\frac{P \overset{\bar{x}}{\rightarrow} C \quad Q \overset{x}{\rightarrow} F}{P | Q \overset{\tau}{\rightarrow} F @ C}$
 - L-PAR_c** $\frac{P \overset{\alpha}{\rightarrow} A}{P | Q \overset{\alpha}{\rightarrow} A | Q}$
 - R-PAR_c** $\frac{Q \overset{\alpha}{\rightarrow} A}{P | Q \overset{\alpha}{\rightarrow} P | A}$
 - RES_c** $\frac{P \overset{\alpha}{\rightarrow} A}{\text{new } x P \overset{\alpha}{\rightarrow} \text{new } x A}$ if $(\alpha \notin \{x, \bar{x}\})$
 - REP_c** $\frac{P | !P \overset{\alpha}{\rightarrow} A}{!P \overset{\alpha}{\rightarrow} A}$

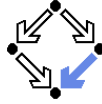
Relationships



- **Structural Congruence Respects Commitment:** If $P \overset{\alpha}{\rightarrow} A$ and $P \equiv Q$, then there exists some B such that $Q \overset{\alpha}{\rightarrow} B$ and $A \equiv B$.
 - Structurally congruent processes have the same commitments.
- **Reaction Agrees with τ -Commitment:** $P \rightarrow P'$ if and only if there exists some P'' such that $P \overset{\tau}{\rightarrow} P''$ and $P'' \equiv P'$.
 - \rightarrow corresponds to the silent commitment $\overset{\tau}{\rightarrow}$ (modulo congruence).

Analogous to the relationships in CCS.

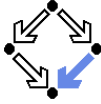
Agent Congruence



- **Agent congruence:** An equivalence relation \simeq on agents is an agent congruence, if the following holds:
 1. If $A \simeq B$, then $\alpha A + M \simeq \alpha B + M$ (for any process M)
 2. If $P \simeq Q$, then $\text{new } a P \simeq \text{new } a Q$, $P|R \simeq Q|R$, $R|P \simeq R|Q$, $!P \simeq !Q$, $\text{new } \vec{x} \langle \vec{y} \rangle . P \simeq \text{new } \vec{x} \langle \vec{y} \rangle . Q$.
 3. If $\{\vec{y}/\vec{x}\}P \simeq \{\vec{y}/\vec{x}\}Q$ for all \vec{y} , then $(\vec{x}).P \simeq (\vec{x}).Q$.

The notion is more complex than process congruence, because messages may be passed in reactions.

Agent Relations

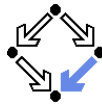


Extend relations on processes to relations on agents.

- **Application of process relation to abstractions resp. concretions:**
Assume binary relation \mathcal{S} on processes, abstractions F, G , concretions C, D .
 - **FSG:**
 $F\langle \vec{y} \rangle \mathcal{S} G\langle \vec{y} \rangle$, for all \vec{y} .
 - **CSD:**
 PSQ , for some processes P and Q and \vec{z}, \vec{y} such that $C \equiv \text{new } \vec{z} \langle \vec{y} \rangle . P$ and $D \equiv \text{new } \vec{z} \langle \vec{y} \rangle . Q$.

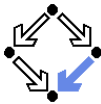
Abstractions are only related, if all their instantiations are related!

Strong Equivalence



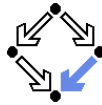
- **Strong simulation:** A binary relation \mathcal{S} on processes is a strong simulation, if PSQ implies
 - if $P \overset{\alpha}{\rightarrow} A$, then there exists some B such that $Q \overset{\alpha}{\rightarrow} B$ and ASB .
- **Strong bisimulation:** A binary relation \mathcal{S} on processes is a strong bisimulation, if both \mathcal{S} and its converse are a strong simulation.
- **Strong equivalence:** Two agents A and B are strongly equivalent, written $A \sim B$, if ASB , for some strong bisimulation \mathcal{S} .
- **Theorem:** strong equivalence \sim is an agent congruence.
 - Not a general congruence.
 - $P \simeq Q$ does not imply $z(y).P \simeq z(y).Q$:
 - $\vec{x}|y \sim \vec{x}.y + y.\vec{x}$.
 - $\{x/y\}\vec{x}|y \not\sim \{x/y\}\vec{x}.y + y.\vec{x}$.

Basic Properties of Replication



Strong equivalence does not imply structural congruence.

- **Propositions:**
 - $!P \sim !P | !P$.
 - $!!P \sim !P$.
 - But $!P \not\equiv !P!P$ and $!!P \not\equiv !P$.
- **Definition:** An agent A is *negative* on x , if its only free occurrences of x are in the form $\vec{x}C$.
 - Cannot receive messages on x or transmit x as a message.
 - If P is negative and $P \overset{\alpha}{\rightarrow} A$, then A is negative on x .
- **Theorems:** Let P, P_1, P_2, F be negative on x .
 1. $\text{new } x (P_1 | P_2 | !xF) \sim \text{new } x (P_1 | !xF) | \text{new } x (P_2 | !xF)$
 2. $\text{new } x (!P | !xF) \sim !\text{new } x (P | !xF)$
 - P_1 and P_2 cannot communicate on x .
 - $\text{new } x (_ | !xF)$ can be pushed inside composition and replication.

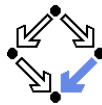


1. Strong Equivalence

2. Observation Equivalence

3. Extensions of the π -Calculus

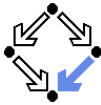
Weak Simulations



- **Weak Simulation:** a binary relation \mathcal{S} on processes is a *weak simulation* if, whenever PSQ , then:
 - if $P \Rightarrow P'$, then there exists Q' such that $Q \Rightarrow Q'$ and $P'SQ'$;
 - if $P \xrightarrow{x(\vec{y})} P'$, then there exists Q' such that $Q \xrightarrow{x(\vec{y})} Q'$, and $P'SQ'$;
 - if $P \xrightarrow{\bar{x}} C$, then there exists D such that $Q \xrightarrow{\bar{x}} D$ and CSD .
- **Proposition:** a binary relation \mathcal{S} on processes is a weak simulation if, whenever PSQ , then:
 - if $P \rightarrow P'$, then there exists Q' such that $Q \Rightarrow Q'$ and $P'SQ'$;
 - if $P \xrightarrow{x(\vec{y})} P'$, then there exists Q' such that $Q \xrightarrow{x(\vec{y})} Q'$, and $P'SQ'$;
 - if $P \xrightarrow{\bar{x}} C$, then there exists D such that $Q \xrightarrow{\bar{x}} D$ and CSD .

To establish that \mathcal{S} is a weak simulation, it is only necessary to check single transitions.

Experiments

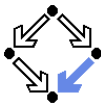


In CCS, an atomic experiment to distinguish process expressions is represented by the relation $\xrightarrow{\lambda}$ ($\lambda \in \{a, \bar{a}\}$, for some action a).

- **Transition relation $\xrightarrow{x(\vec{y})}$:**
 $P \xrightarrow{x(\vec{y})} P'$ iff, for some F , $P \xrightarrow{x} F$ and $F(\vec{y}) \equiv P'$.
- **Transition relation $\xrightarrow{x(\vec{y})}$ (the atomic input experiments):**
 $P \xrightarrow{x(\vec{y})} P'$ iff, for some P_0, P_1 , $P \Rightarrow P_0 \xrightarrow{x(\vec{y})} P_1 \Rightarrow P'$
- **Transition relation $\xrightarrow{\bar{x}}$ (the atomic output experiments):**
 $P \xrightarrow{\bar{x}} \text{new } \vec{z} \langle \vec{y} \rangle . P'$ if, for some P_0, P'' ,
 $P \Rightarrow P_0 \xrightarrow{\bar{x}} \text{new } \vec{z} \langle \vec{y} \rangle . P''$, and $P'' \Rightarrow P'$.

Two relations required for capturing input and output experiments.

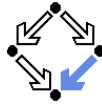
Observation Equivalence



- **Weak Bisimulation:** a binary relation \mathcal{S} on processes is a weak bisimulation, if both \mathcal{S} and its converse are weak simulations.
- **Observation Equivalence:** two agents A and B are observation equivalent, written $A \simeq B$, if ASB , for some weak bisimulation \mathcal{S} .
- **Propositions:**
 - $P \sim Q$ implies $P \simeq Q$
 - $P \simeq \tau.P$
- **Theorem:** observation equivalence is an agent congruence.

Processes are not observationally equivalent only if they can be distinguished by some input or output experiment.

Unique Solution of Equations



- **Theorem:** Let $\vec{X} = X_1, X_2, \dots$ be a (possibly infinite) sequence of variables over abstractions and let \vec{F} and \vec{G} be two solutions of the system of equations

$$X_1(\vec{x}_1) \simeq \alpha_{11}A_{11} + \dots + \alpha_{1n_1}A_{1n_1}$$

$$X_2(\vec{x}_2) \simeq \alpha_{21}A_{21} + \dots + \alpha_{2n_2}A_{2n_2}$$

...

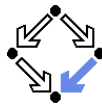
where each term αA on the right sides takes one of the forms

- xH where H is an abstraction $(\vec{v}).X_k(\vec{w})$
- $\bar{x}C$ where C is a concretion new $\vec{u}(\vec{v}).X_k(\vec{w})$

Then $F_i \simeq G_i$, for all i .

A system of agents is uniquely (up to observation equivalence) defined by a mutually recursive set of agent equations.

A Type System for the π -Calculus



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

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

$$Trans\langle talk_2, switch_2, gain_2, lose_2 \rangle = \\ talk_2 . Trans\langle talk_2, switch_2, gain_2, lose_2 \rangle + \\ lose_2(t, s) . switch_2\langle t, s \rangle . Idtrans\langle gain_2, lose_2 \rangle.$$

- Channel type $lose_1 : CHAN(\tau, \sigma)$
 - Output action $\overline{lose_1}\langle talk_2, switch_2 \rangle$
 - Message types $talk_2 : \tau, switch_2 : \sigma$
- Channel type $switch_2 : CHAN(\tau, \sigma)$
 - Output action $switch_2\langle talk_1, switch_1 \rangle$
 - Message types $talk_1 : \tau, switch_1 : \sigma$
- Consequence $\sigma = CHAN(\tau, \sigma)$
 - A channel may carry messages of its own type.

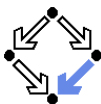
No hierarchy of channel types possible, we need another approach.

1. Strong Equivalence

2. Observation Equivalence

3. Extensions of the π -Calculus

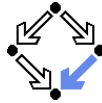
Sorts



- Collection of sorts Σ
 - $x : \sigma \dots$ to name x , sort σ is assigned.
 - To every name, a sort is assigned.
 - Every sort is assigned to infinitely many names.
- Set of sort lists Σ^*
 - $\vec{x} : \vec{\sigma} \dots$ to name sequence \vec{x} , sort list $\vec{\sigma}$ is assigned.
 - $\vec{x} = x_1, \dots, x_n, \vec{\sigma} = \sigma_1, \dots, \sigma_n.$
 - $x_i : \sigma_i, 1 \leq i \leq n.$
 - $F : \vec{\sigma}$
 - Abstraction $F = (\vec{x})P$
 - $\vec{x} : \vec{\sigma}$

Sorts shall ensure that names are used properly by processes.

Sortings



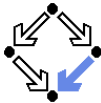
Take a set Σ of sorts.

- A **sorting** $ob : \Sigma \rightarrow \Sigma^*$ over Σ :
 - Partial function mapping sorts to sort sequences.
- A process (family) **respects** a sorting ob :
 - For every subterm $x(\vec{y}).P$ or $\bar{x}(\vec{y}).P$ of the process (family):
if $x : \sigma$, then $\vec{y} : ob(\sigma)$.
- **Example:** $\Sigma = \{\text{TALK, SWITCH, GAIN, LOSE}\}$
 - $talk_i : \text{TALK}, switch_i : \text{SWITCH}, gain_i : \text{GAIN}, lose_i : \text{LOSE}$
 - This sorting is (the only one) respected by the mobile phone system:

$$ob : \begin{cases} \text{TALK} \mapsto \epsilon \\ \text{SWITCH} \mapsto \text{TALK, SWITCH} \\ \text{GAIN} \mapsto \text{TALK, SWITCH} \\ \text{LOSE} \mapsto \text{TALK, SWITCH} \end{cases}$$

- We have another respected sorting for $\Sigma = \{\text{TALK, SWITCH}\}$ (which?)

Sorting and Process Relations



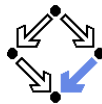
- **Proposition:** Structural congruence and reaction preserve sortings:
 - If $P \equiv P'$ or $P \rightarrow P'$ and P respects ob , then P' respects ob .
 - Sorting thus constrains the pattern of interaction.
- **Example:** Process $P|Q$ with free names $x : A, y : B, u : C, v : D$
 - Process respects the sorting

$$ob : \begin{cases} A \mapsto B \\ B \mapsto \epsilon \\ C \mapsto D \\ D \mapsto \epsilon \end{cases}$$

- Assume P contains only x free.
- Then P can receive y from Q and use it but it cannot receive u or v .

Sorting assists in analyzing behavior.

Processes as Messages

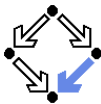


Why only allow names as messages?

- Extend action prefixes π to transmit **processes**:
 - Allow **process variables** p, q, r, \dots in input actions.
 - Allow **process expressions** in output actions.
- **Example:** process R is transmitted.
 - $P = \bar{x}\langle R \rangle.P'$.
 - $Q = x(r).(r|R|Q')$.
 - $P|Q \rightarrow P'|R|R|Q'$.
- **Translation:**
 - $\hat{P} = (\text{new } z)\bar{x}\langle z \rangle.(!z.R | P')$.
 - $\hat{Q} = x(z).(\bar{z}\langle z \rangle|Q')$.
 - $\hat{P}|\hat{Q} \Rightarrow (\text{new } z)!z.r|P'|R|R|Q' \sim 0|P'|R|R|Q' \equiv P'|R|R|Q'$.

The higher-order π -calculus can be translated into the plain calculus.

Abstractions as Messages



Transmitted processes may be also parametric.

- **Example:** abstraction F is transmitted.
 - $P = \bar{x}\langle F \rangle.P'$.
 - $Q = x(f).(f\langle u \rangle|f\langle v \rangle|Q')$.
 - $P|Q \rightarrow P'|F\langle u \rangle|F\langle v \rangle|Q'$.
- **Translation:**
 - $\hat{P} = (\text{new } z)\bar{x}\langle z \rangle.(!z.F | P')$.
 - $\hat{Q} = x(z).(\bar{z}\langle u \rangle|\bar{z}\langle v \rangle|Q')$.
 - $\hat{P}|\hat{Q} \Rightarrow \dots \sim \dots \equiv P'|F\langle u \rangle|F\langle v \rangle|Q'$.

Translation takes correctly care of name scopes (free names of abstraction are bound in sender, free names of arguments are bound in receiver).