

The linear π -calculus

Mauro Piccolo

February 16, 2007

- VY06 D. Varacca , N. Yoshida *Typed Event structure and the π -calculus* In MFPS, 2006
- YHB03 N. Yoshida, K. Honda, M. Berger *Strong Normalization and the π -calculus* Journal of Information and Computation, 2003
- YHB01 N. Yoshida, K. Honda, M. Berger *Sequentiality and the π -calculus* In Proc. of TLCA'01, 2001
- San95 D. Sangiorgi *Internal Mobility and Agent passing calculi* In Proc. ICALP'95, 1995
- SW01 D. Sangiorgi , D. Walker *The π -calculus: a Theory of Mobile Processes* Cambridge University Press, 2001
- Bor98 M. Boreale *On the expressivness of Internal Mobility in Name-Passing calculi* Theoretical Computer Science, 1998

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Concurrent system

- a set of processes (agents) that interact together
 - ◆ concurrent computation
 - ◆ synchronization
 - ◆ mobility
- each process is defined by specifying all its possible interaction with the rest of the system

How do we describe a such complex reality?

References

Introduction

The π -calculus

From π -calculus to
linear π -calculus

The π -calculus

References

Introduction

The π -calculus

Syntax

Examples:

Operational
semantics

Reduction rules

External vs internal
mobility

Example

Actions - notation

Labelled Transition

System - 1

Labelled Transition

System - 2

Labelled Transition

System - 3

Behavioural

equivalence - 1

Behavioural

equivalence - 2

Discussion

From π -calculus to
linear π -calculus

Names and variables:

a, b, c (names) , x, y, z , (var.) s, t (both)

$\pi ::= a(\vec{x})$ input

Action capabilities:

$\bar{a}\langle\vec{t}\rangle$ output

τ internal action

$P ::= \pi.P$ prefixed action

$\sum_{i \in I} \pi_i.P_i$ prefixed sum

$P|P$ parallel composition

Processes:

νxP restriction

$!P$ replication

0 termination

$a(x).P$ and νxP are binding constructs (the occurrences of x are bounded in P)

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

input and output actions

$$P = a(x).\bar{x}\langle t \rangle.P'$$

parallel execution

$$Q = a(x).Q_1|\bar{v}\langle t \rangle.Q_2$$

possible interaction

$$S = a(x).S|\bar{a}\langle t \rangle.S'$$

restriction (scope extrusion)

$$R = \nu z(\bar{u}\langle z \rangle.z(w).R')$$

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

Behaviour of a process

- internal dynamics of a process
 - ◆ reduction rules
- interaction with the environment
 - ◆ labelled transition system

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

The **structural congruence** (\equiv) is the smallest contextual congruence satisfying the following axioms

α -conversion of bound names

$$P|Q \equiv Q|P$$

$$\nu x \nu y P \equiv \nu y \nu x P$$

$$\nu z (P|Q) \equiv (\nu z P)|Q \text{ if } z \notin fn(Q)$$

$$P|(Q|R) \equiv (P|Q)|R$$

$$P|0 \equiv P$$

$$\nu z 0 \equiv 0$$

$$!P \equiv P|!P$$

The **reduction** (\longrightarrow) is defined by the following rules

$$a(\vec{x}).P + M|\bar{a}\langle\vec{t}\rangle.Q + N \longrightarrow P\{\vec{t}/\vec{x}\}|Q \quad (1)$$

$$\tau.P + M \longrightarrow P \quad (2)$$

$$P \longrightarrow P' \Rightarrow P|Q \longrightarrow P'|Q \quad (3)$$

$$P_1 \equiv Q_1 \longrightarrow Q_2 \equiv P_2 \Rightarrow P_1 \longrightarrow P_2 \quad (4)$$

$$P \longrightarrow P' \Rightarrow \nu x P \longrightarrow \nu x P' \quad (5)$$

References
Introduction

The π -calculus

Syntax

Examples:
Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation
Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

External vs internal mobility

External mobility: communication of free names

- Example: $P = \bar{c}\langle a \rangle.P'$

Internal mobility: communication of private (fresh) names

- Example: $P = \nu z(\bar{a}\langle z \rangle.P')$
- the source of the expressive power of the π -calculus [Sangiorgi95]

References
Introduction

The π -calculus

Syntax

Examples:
Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation
Labelled Transition System - 1

Labelled Transition System - 2

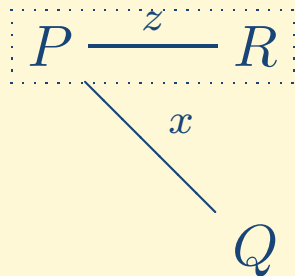
Labelled Transition System - 3

Behavioural equivalence - 1

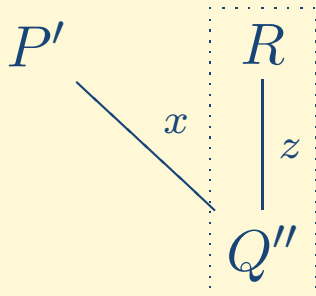
Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus



If $\nu z(P|R)|Q$ and if $P = \bar{x}\langle z\rangle.P'$ with z non-free in P' e $Q = x(y).Q'$ then



$P'|\nu z(Q''|R)$ with $Q'' = Q'\{z/y\}$

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

Actions - notation

$\alpha ::=$	$\bar{x}\langle y \rangle$	free output	xy	input	
	$ \nu z \bar{x}\langle z \rangle$	bound output	τ	internal	
we write	$\bar{x}(z) =_{def} \nu z \bar{x}\langle z \rangle$				
α	$subj(\alpha)$	$obj(\alpha)$	$fn(\alpha)$	$bn(\alpha)$	$n(\alpha)$
$\bar{x}\langle y \rangle$	x	y	$\{x, y\}$	\emptyset	$\{x, y\}$
xy	x	y	$\{x, y\}$	\emptyset	$\{x, y\}$
$\bar{x}(z)$	x	z	$\{x\}$	$\{z\}$	$\{x, z\}$
τ	—	—	\emptyset	\emptyset	\emptyset

- References
- Introduction
- The π -calculus
- Syntax
- Examples:
- Operational semantics
- Reduction rules
- External vs internal mobility
- Example
- Actions - notation**
- Labelled Transition System - 1
- Labelled Transition System - 2
- Labelled Transition System - 3
- Behavioural equivalence - 1
- Behavioural equivalence - 2
- Discussion
- From π -calculus to linear π -calculus

Labelled Transition System - 1

$$\frac{}{u(x).P \xrightarrow{uy} P\{y/x\}} \text{ in } \frac{}{\bar{u}\langle x \rangle.P \xrightarrow{\bar{u}\langle x \rangle} P} \text{ out } \frac{}{\tau.P \xrightarrow{\tau} P} \text{ tau}$$

$$\frac{P \xrightarrow{\alpha} P' \quad \text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q} \text{ p-l} \quad \frac{P \xrightarrow{\bar{x}\langle y \rangle} P' \quad Q \xrightarrow{xy} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \text{ c}$$

$$\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'[\text{sum-l}]}$$

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

Labelled Transition System - 2

$$\frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{xz} Q'}{P|Q \xrightarrow{\tau} \nu z(P'|Q')} \text{ cl - l} \quad \frac{P \xrightarrow{\bar{x}(z)} P'}{\nu z P \xrightarrow{\bar{x}(z)} P'} \text{ open}$$

$$\frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{\nu z P \xrightarrow{\alpha} \nu z P'} \text{ res} \quad \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' | !P} \text{ r - act}$$

$$\frac{P \xrightarrow{\bar{x}(y)} P' \quad P \xrightarrow{xy} P''}{!P \xrightarrow{\tau} (P'|P'') | !P} \text{ r - c} \quad \frac{P \xrightarrow{\bar{x}(z)} P' \quad P \xrightarrow{xz} P'' \quad z \notin fn(P)}{!P \xrightarrow{\tau} (\nu z(P'|P'')) | !P}$$

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

Harmony lemma:

- $P \equiv \xrightarrow{\alpha} P' \Rightarrow P \xrightarrow{\alpha} \equiv P'$
- $P \longrightarrow P' \iff P \xrightarrow{\tau} \equiv P'$

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

Strong bisimilarity: Strong bisimilarity is the largest binary symmetric relation \sim on processes such that whenever $P \sim Q$, if $P \xrightarrow{\alpha} P'$ then there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \sim Q'$.

Weak transition relations

- \Longrightarrow is the reflexive and transitive closure of $\xrightarrow{\tau}$
- $\xRightarrow{\alpha}$ is $\Longrightarrow \xrightarrow{\alpha} \Longrightarrow$, where α is an action.

Weak bisimilarity: weak bisimilarity is the largest binary symmetric relation \approx on processes such that whenever $P \approx Q$, if $P \xrightarrow{\alpha} P'$ then there exists Q' such that $Q \xRightarrow{\alpha} Q'$ and $P' \approx Q'$

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

Properties:

- $\sim \subseteq \approx$
- \approx is an equivalence relation

BUT

They are not contextual congruence relations

- let $P = \bar{x}|y$ and $Q = \bar{x}.y + y.\bar{x}$ be two processes
- let $C[] = a(x).[]$ be a context

we have

- $P \sim Q$ but $C[P] \not\sim C[Q]$

References
Introduction

The π -calculus

Syntax

Examples:
Operational semantics

Reduction rules
External vs internal mobility

Example

Actions - notation
Labelled Transition System - 1

Labelled Transition System - 2

Labelled Transition System - 3

Behavioural equivalence - 1

Behavioural equivalence - 2

Discussion

From π -calculus to linear π -calculus

π -calculus

- it models concurrent mobile computation
- it has an intuitive notion of observational equivalence (bisimilarity)

But

- reduction is in general not confluent
- there is no canonical normal form
- bisimilarity is not a contextual equivalence relation

Solutions:

- restricting the calculus (this talk)
- extending the calculus
 - ◆ Fusion calculus (Parrow Victor)
 - ◆ Solos (Laneve Parrow Victor)

References

Introduction

The π -calculus

Syntax

Examples:

Operational semantics

Reduction rules

External vs internal mobility

Example

Actions - notation

Labelled Transition

System - 1

Labelled Transition

System - 2

Labelled Transition

System - 3

Behavioural

equivalence - 1

Behavioural

equivalence - 2

Discussion

From π -calculus to linear π -calculus

From π -calculus to linear π -calculus

References

Introduction

The π -calculus

**From π -calculus to
linear π -calculus**

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

$\lambda\pi$ -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type
environments

Composition of type
environments

Typing rules

Adding constraints

We add the following restrictions:

Asynchrony: the act of sending a datum and the act of receiving it are separate. In particular no process need to wait to send a datum.

Locality: channels received in input are only used for output and dually channels sent in output are only used for input

Internal mobility: the communication involves only private (fresh) names

Linearity: The linearity constraint gives a discipline on how much a process can use a name.

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

λ π -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type

environments

Composition of type

environments

Typing rules

Adding constraints

Syntax:

$$P ::= a(\vec{x}).P \mid \bar{a}\langle\vec{t}\rangle \mid P \mid P \mid \nu\vec{x}P \mid !P \mid 0$$

Reduction:

$$a(\vec{x}).P \mid \bar{a}\langle\vec{t}\rangle \longrightarrow P\{\vec{t}/\vec{x}\}$$

Output is not blocking and can be sequentialized:

$$\nu y, z(\bar{x}\langle y\rangle \mid \bar{y}\langle z\rangle \mid \bar{z}\langle a\rangle \mid R)$$

with $y, z \notin fn(R)$

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous π -calculus

Expressing

synchrony in

λ - π -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type environments

Composition of type environments

Typing rules

Adding constraints

Expressing synchrony in $A\pi$ -calculus

The translation $\llbracket \cdot \rrbracket$ from π to $A\pi$ is an homomorphism except that

$$\llbracket \bar{a}\langle \vec{t} \rangle . P \rrbracket = \nu y (\bar{a}\langle \vec{t}, y \rangle | y(). \llbracket P \rrbracket)$$

$$\llbracket a(\vec{x}) . P \rrbracket = a(\vec{x}, y) . (\bar{y}\langle \rangle | \llbracket Q \rrbracket)$$

The transition $\bar{a}\langle \vec{t} \rangle . P | a(\vec{x}) . Q \xrightarrow{\tau} P | Q\{\vec{t}/\vec{x}\}$ is mimicked by the sequence

$$\begin{aligned} \llbracket \bar{a}\langle \vec{t} \rangle . P | a(\vec{x}) . Q \rrbracket &\equiv \nu y ((\bar{a}\langle \vec{t}, y \rangle | y(). \llbracket P \rrbracket) | a(\vec{x}, z) . (\bar{z}\langle \rangle | \llbracket Q \rrbracket)) \\ &\longrightarrow \nu y (y(). \llbracket P \rrbracket | (\bar{z}\langle \rangle | \llbracket Q \rrbracket)\{\vec{t}/\vec{x}, y/z\}) \\ &\equiv \nu y (y(). \llbracket P \rrbracket | (\bar{y}\langle \rangle | \llbracket Q \rrbracket)\{\vec{t}/\vec{x}\}) \\ &\longrightarrow \llbracket P \rrbracket | \llbracket Q\{\vec{t}/\vec{x}\} \rrbracket \end{aligned}$$

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous π -calculus

Expressing synchrony in $A\pi$ -calculus

Asynchronous

Localized π -calculus

Private Localized π -calculus

Linear π -calculus [YoshidaHondaBerger03]

Linearity

Types and type environments

Composition of type environments

Typing rules

Adding constraints

Syntax:

$$P ::= a(\vec{x}).P \mid \bar{a}\langle\vec{t}\rangle \mid P|P \mid \nu\vec{x}P \mid !P \mid 0$$

where we impose that in the process $a(\vec{x}).P$ the names \vec{x} cannot occur free in P as a subject of an input prefix. (**Output capability constraint**)

Locality: Every process that receive via a name is local to the process that create that name.

Example: a printer manager process can communicate to another process the capability to send a job to the printer but not the capability to receive a job for the printer

Expressiveness : the same as $A\pi$ ([Boreale98])

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions
Asynchronous
 π -calculus
Expressing
synchrony in
 $A\pi$ -calculus

**Asynchronous
Localized π -calculus**

Private Localized
 π -calculus
Linear π -calculus
[YoshidaHondaBerger03]

Linearity

Types and type
environments

Composition of type
environments

Typing rules

Adding constraints

Only internal mobility is allowed

Syntax: we pose $\bar{a}(\vec{x}).P =_{def} \nu \vec{x}(\bar{a}\langle \vec{x} \rangle.P)$

$P ::=$	$a(\vec{x}).P$	input
	$\bar{a}(\vec{x}).P$	output
	$P P$	parallel composition
	$\nu \vec{x} P$	restriction
	$K\langle \vec{x} \rangle$	recursion
	0	termination

where we impose that

- in $\bar{a}(\vec{x}).P$ the names \vec{x} can occur in P only as the subject of *input* prefixes.
- in $a(\vec{x}).P$ the names \vec{x} can occur in P only as the subject of an *output*.

Reduction: $a(\vec{x}).P|\bar{a}(\vec{x}).Q \longrightarrow \nu \vec{x}P|Q$

Expressivness: The same as $LA\pi$ -calculus ([Boreale98])

References

Introduction

The π -calculus

From π -calculus to
linear π -calculus

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

$A\pi$ -calculus

Asynchronous

Localized π -calculus

Private Localized
 π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type

environments

Composition of type

environments

Typing rules

Adding constraints

Linear π -calculus [YoshidaHondaBerger03]

Syntax: we pose $\bar{a}(\vec{x}) P =_{def} \nu \vec{x}(\bar{a}(\vec{x})|P)$

$P ::=$ $a(\vec{x}).P$ input
 $\bar{a}(\vec{x}) P$ asynchronous output
 $!a(\vec{x}).P$ guarded replication
 \dots and the same as above

structural and reduction rules:

$$\bar{x}(\vec{u}) \bar{y}(\vec{v}) P \equiv \bar{y}(\vec{v}) \bar{x}(\vec{u}) P \text{ se } x \notin \vec{v} \text{ e } y \notin \vec{u}$$

$$\bar{x}(\vec{u}) (P|Q) \equiv (\bar{x}(\vec{u}) P)|Q \text{ se } \vec{u} \notin Q$$

$$\nu y \bar{x}(\vec{u}) P \equiv \bar{x}(\vec{u}) \nu y P \text{ se } y \notin \{x, \vec{u}\}$$

$$a(\vec{x}).P|\bar{a}(\vec{x}) Q \longrightarrow \nu \vec{x}(P|Q)$$

$$!a(\vec{x}).P|\bar{a}(\vec{x}) Q \longrightarrow !a(\vec{x}).P|\nu \vec{x}(P|Q)$$

$$P \longrightarrow P' \Rightarrow \bar{a}(\vec{x}) P \longrightarrow \bar{a}(\vec{x}) P'$$

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

λ π -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type environments

Composition of type environments

Typing rules

Adding constraints

Action and channel modes

■ input modes

- ↓ linear (ex. $a(x).P$)
- ! replicated (ex. $!a(x).P$)

■ output modes (ex $\bar{a}(x) P$)

- ↑ linear
- ? replicated

■ neutral (\updownarrow)

We use pO (resp. pI) to vary on output (resp. input) modes

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

$\lambda\pi$ -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type environments

Composition of type environments

Typing rules

Adding constraints

Types and type environments

$$\begin{aligned}\tau & ::= \tau_I | \tau_O | \downarrow & \tau_I & ::= (\vec{\tau}_O)^{pI} & \tau_O & ::= (\vec{\tau}_I)^{pO} \\ \Gamma & ::= a_1 : \tau_1, \dots, a_n : \tau_n\end{aligned}$$

- Γ is always well formed (each name appear only once)
- $Dom(\Gamma)$ is the set of names that occur in Γ . If $\Gamma = \Delta, a : \tau$ we note $\Gamma(a) = \tau$
- Given a type τ , we define its **dual** τ^\perp in the following way

- ◆ $((\vec{\tau}_I)^\downarrow)^\perp = (\vec{\tau}_I^\perp)^\uparrow$
- ◆ $((\vec{\tau}_I)^\?)^\perp = (\vec{\tau}_I^\perp)^\!$
- ◆ $((\vec{\tau}_O)^\uparrow)^\perp = (\vec{\tau}_O^\perp)^\downarrow$
- ◆ $((\vec{\tau}_O)^\!)^\perp = (\vec{\tau}_O^\perp)^\?$

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

λ - π -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type environments

Composition of type environments

Typing rules

Adding constraints

Composition of type environments

Let Γ and Δ two type environments such that, for all $a \in Dom(\Gamma) \cap Dom(\Delta)$ we have that one of the following holds

- $\Gamma(a) = \Delta(a)^\perp$
- $\Gamma(a) = \Delta(a) = (\vec{\tau}_I)^?$

Then we define $\Gamma \odot \Delta$ the environment Ξ such that $Dom(\Xi) = Dom(\Gamma) \cup Dom(\Delta)$ and

$$\Xi(a) = \begin{cases} \Gamma(a) & \text{if } a \in Dom(\Gamma) \setminus Dom(\Delta) \\ \Delta(a) & \text{if } a \in Dom(\Delta) \setminus Dom(\Gamma) \\ \updownarrow & \text{if } \Gamma(a) = \Delta(a)^\perp \text{ with a linear modality} \\ (\tau_O)! & \text{if } \Gamma(a) = \Delta(a)^\perp \text{ with a replicated modality} \\ (\tau_I)^? & \text{if } \Gamma(a) = \Delta(a) = (\tau_I)^? \end{cases}$$

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

$\lambda\pi$ -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type environments

Composition of type environments

Typing rules

Adding constraints

Typing rules

$$\frac{}{0 \triangleright} \text{zero} \quad \frac{P_1 \triangleright \Gamma_1 \quad P_2 \triangleright \Gamma_2}{P_1 | P_2 \triangleright \Gamma_1 \odot \Gamma_2} \text{par} \quad \frac{P \triangleright \Gamma, x : \tau \quad md(\tau) \in \{\uparrow, \downarrow, !\}}{\nu x P \triangleright \Gamma}$$

$$\frac{P \triangleright \Gamma \quad md(\tau) \in \{\uparrow, ?\}}{P \triangleright \Gamma, x : \tau} \text{weak} \quad \frac{P \triangleright \vec{x} : \vec{\tau}_O, \downarrow \Gamma_1, ?\Gamma_2}{a(\vec{x}).P \triangleright \uparrow \Gamma_1, ?\Gamma_2, a : (\vec{\tau}_O)^\downarrow} \text{in}^\downarrow$$

$$\frac{P \triangleright \vec{x} : \vec{\tau}_O, ?\Gamma}{!a(\vec{x}).P \triangleright ?\Gamma, a : (\vec{\tau}_O)^\downarrow} \text{in}^\downarrow \quad \frac{P \triangleright \Gamma, \vec{x} : \vec{\tau}_I}{\bar{a}(\vec{x}) P \triangleright \Gamma \odot (a : (\vec{\tau}_I)^{pO})} \text{out}$$

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

A π -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHonda

Berger03]

Linearity

Types and type

environments

Composition of type

environments

Typing rules

Adding constraints

The typing system above guarantees the confluence property (\longrightarrow is Church-Rosser).

We can add constraints:

acyclicity: strong normalization
[YoshidaHondaBerger03]

sequentiality: full abstraction in PCF
[YoshidaHondaBerger01]

We can also extend the calculus with:

branching/selection: additives

non determinism: see [VaraccaYoshida06]

References

Introduction

The π -calculus

From π -calculus to linear π -calculus

Restrictions

Asynchronous

π -calculus

Expressing

synchrony in

λ π -calculus

Asynchronous

Localized π -calculus

Private Localized

π -calculus

Linear π -calculus

[YoshidaHondaBerger03]

Linearity

Types and type

environments

Composition of type

environments

Typing rules

Adding constraints