Programming Web Services with Choreography

Marco Carbone
IT University of Copenhagen

FLACOS'09, Toledo, Spain
--- September 2009

This Talk In One Slide

- **Choreography and Web Services (07)**
  - A model for Choreography
  - Session Types
  - Global Calculus
  - About End-Point Projection (EPP)
- **Interactional Exceptions and Choreography (08&workinprogress)**
  - Session Types and Exceptions
  - Extension of the Global Calculus
  - About End-Point Projection (EPP)

Choreography?

- Choreography as a way of describing communication-based systems focusing on global message flows
- Idea from WS-CDL, the *Web Services Choreography Description Language*
- XML-based description language
- Developed by W3C (since 2003) in collaboration with private companies Pi4Tech, Adobe, Oracle, Sun, etc.
- π-calculus experts invited since 2004 (R. Milner and us)

Choreography?

- Choreography as a way of describing communication-based systems focusing on global message flows
- Idea from WS-CDL, the *Web Services Choreography Description Language*
- XML-based description language
- Developed by W3C (since 2003) in collaboration with private companies Pi4Tech, Adobe, Oracle, Sun, etc.
- π-calculus experts invited since 2004 (R. Milner and us)

Joint with K. Honda & N. Yoshida

IT University of Copenhagen

Choreography? (2)

- Choreography as a way of describing communication-based systems focusing on global message flows

  1. Entities have a common goal
  2. No single point of control
  3. Only describe **communications**

"Dancers dance following a global scenario without a single point of control", WS-CDL working group

Example: End Points

ALICE

send Bob<“Hello”>;
receive (z)

Example: End Points

ALICE

send Bob<“Hello”>;
receive (z)

BOB

receive (x)
send Carl<“Hello”>;

Example: End Points

ALICE

send Bob<“Hello”>;
receive (z)

BOB

receive (x)
send Carl<“Hello”>;

CARL

receive (y)
send Alice<“Hello”>;

Example: End Points

ALICE

send Bob<“Hello”>;
receive (z)

BOB

receive (x)
send Carl<“Hello”>;

CARL

receive (y)
send Alice<“Hello”>;

What happens in this system?

- Alice writes to Bob, then
- Bob writes to Carl, then
- Carl writes to Alice

Example: Choreography

Alice → Bob <“Hello”, x> .

Bob → Carl <“Hello”, y> .

Carl → Alice <“Hello”, z>

Example: comparison

ALICE

send Bob<"Hello">;
receive (z)

BOB

receive (x)
send Carl<"Hello">;

CARL

receive (y)
send Alice<"Hello">;

Alice → Bob <"Hello", x> .
Bob → Carl <"Hello", y> .
Carl → Alice <"Hello", z>

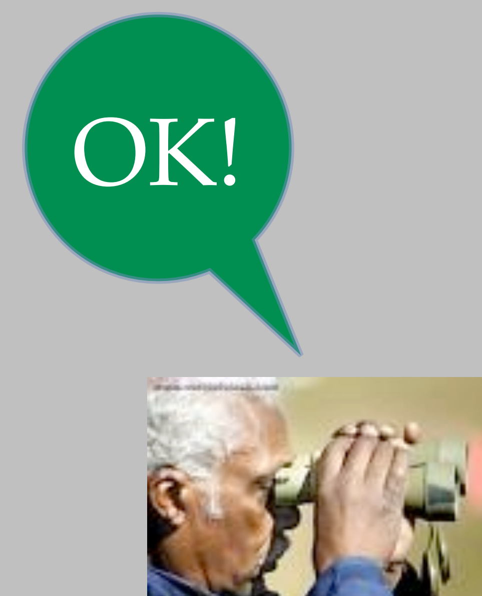
Why Choreography?

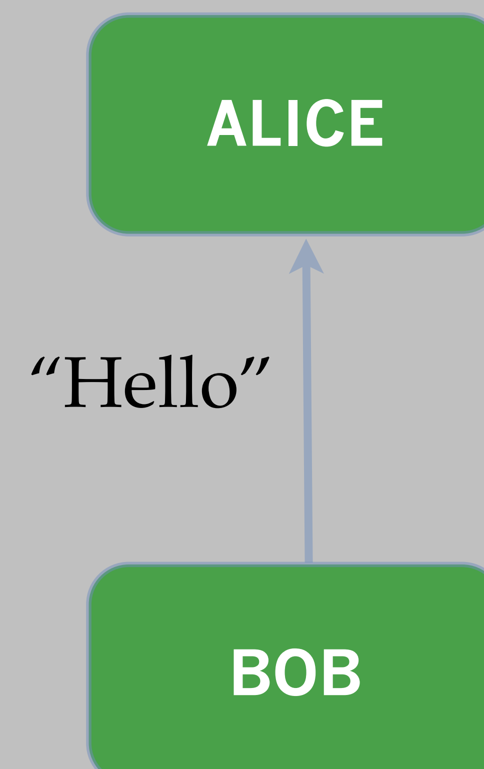
- *Useful* at design stage
- *Abstraction* of a system for formal reasoning
- *Monitoring*

Why Choreography?

- *Useful* at design stage
- *Abstraction* of a system for formal reasoning
- *Monitoring*

Alice → Bob <“Hello”, x> .
Bob → Carl <“Hello”, y> .
Carl → Alice <“Hello”, z>





Why Choreography?

- *Useful* at design stage
- *Abstraction* of a system for formal reasoning
- *Monitoring*

What???
&^%#@

Alice → Bob <“Hello”, x> .
Bob → Carl <“Hello”, y> .
Carl → Alice <“Hello”, z>

“Cheat Carl”

ALICE

BOB

Why Choreography?

- *Useful* at design stage
- *Abstraction* of a system for formal reasoning
- *Monitoring*
- ...

Observation & Question

1. Choreography, ***nice***

2. End-point, directly gives implementation of communication primitives

Observation & Question

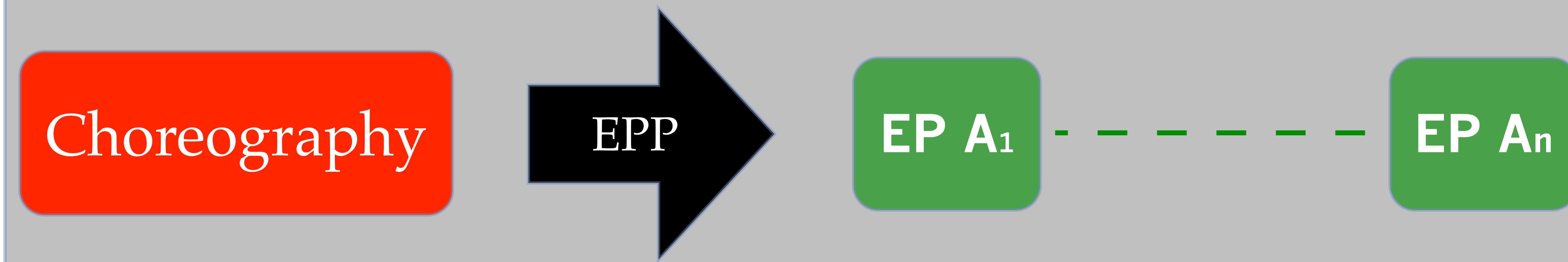
1. Choreography, ***nice***

2. End-point, directly gives implementation of communication primitives

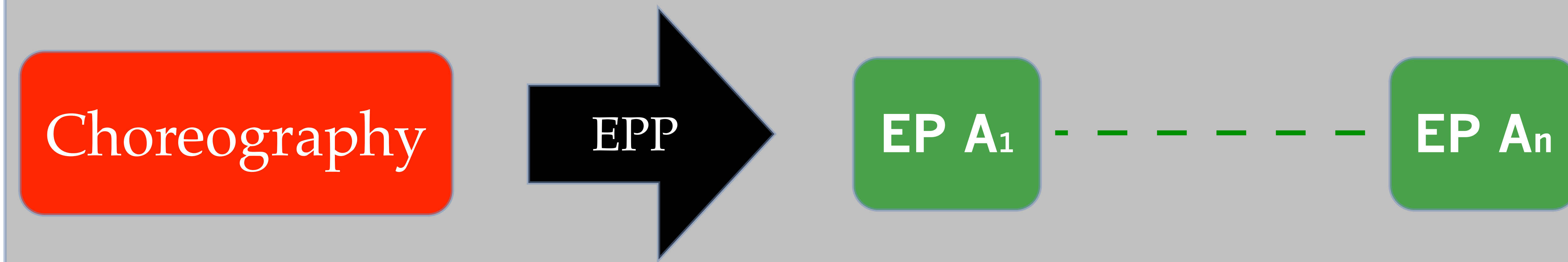
QUESTION

- Can we formally relate 1. and 2.?
  - ➡ Can we define an automated mapping from 1. to 2.?

A possible solution with sessions [ESOP07]

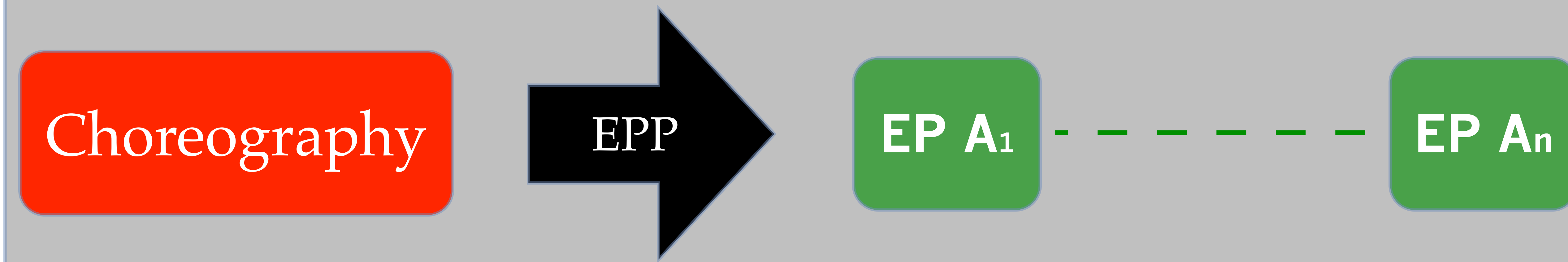


A possible solution with sessions [ESOP07]



- Define two models (process algebras):
  - Global Calculus (GC);
  - End-point Calculus (EPC);

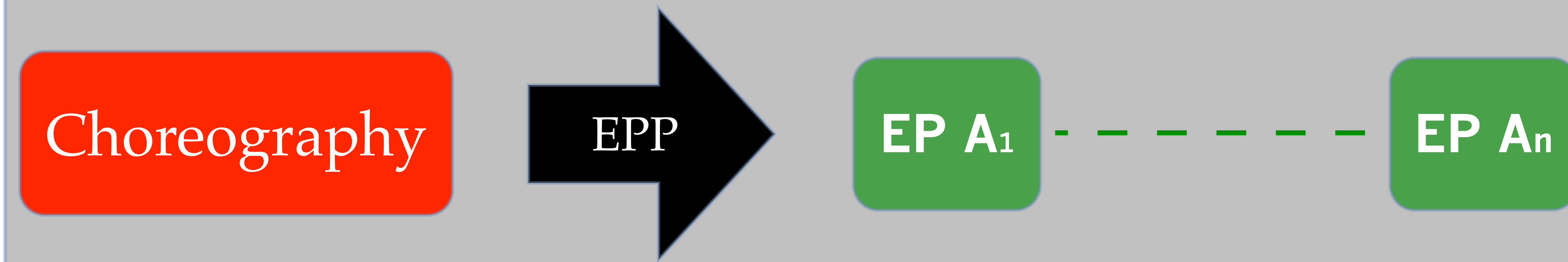
A possible solution with sessions [ESOP07]



- Define two models (process algebras):
  - Global Calculus (GC);
  - End-point Calculus (EPC);

- Define an *efficient* mapping (EPP) from GC to EPC

A possible solution with sessions [ESOP07]



- Define two models (process algebras):
  - Global Calculus (GC);
  - End-point Calculus (EPC);

- Define an *efficient* mapping (EPP) from GC to EPC

- EPP theorem: is EPP always correct?

Related Work

- **Q. Zongyan et al. [WWW07]**
  CCS-like approach (similar to ours), no sessions
- **Bultan et al.**
  (Communicating) Finite State Automata
- **BIP, REO**
- **MSC, Protocol Narrations, etc.**

Sessions (Types) in two slides...

Channel-based communication:

- two types of channels
  - service channels: ch,... (e.g. services, public channels)
  - session channels: s,t,...(e.g. session id's)
- participants invoke shared channels and
- then they communicate over session channels (s)

Sessions (Types) in two slides... (2)

- Each service channel ch has a type α:
    - ch@B : α
- "ch is located at B and is used as α"
- α specifies the direction and the type of each message sent in a session (e.g. over s)
- !(int).?(bool) + !(real).!(real).?(bool)

Service Channel Principle

A service (ch) is **always available in many copies**

IT University
of Copenhagen

Service Channel Principle

A service (ch) is **always available in many copies**

Alice → **Bob** ch(...) .
Bob → **Carl** <"Hello", y> .
...
Carl → **Bob** ch (...). ...

Global Calculus

$I ::=$	$A \to B : ch(s).\ I$	(init)
$\mid$	$A \to B : s<op, e, y>.\ I$	(com)
$\mid$	if $e@A$ then I_1 else I_2	(cond)
$\mid$	$I_1 + I_2$	(sum)
$\mid$	$I_1 \mid I_2$	(par)
$\mid$	$rec\ X.\ I$	(rec)
$\mid$	X	(recVar)
$\mid$	$(\nu s)\ I$	(res)

Global Calculus

I ::=	A → B : ch(s). I	(init)
	| A → B : s<op, e,y>. I	(com)
	| if e@A then I_1 else I_2	(cond)
	| $I_1 + I_2$	(sum)
	| $I_1 \mid I_2$	(par)
	| rec X. I	(rec)
	| X	(recVar)
	| (νs) I	(res)

Global Calculus

$I ::= A \rightarrow B : ch(s).\ I$ (init)

$\mid A \rightarrow B : s\langle op, e, y\rangle.\ I$ (com)

$\mid$ if $e@A$ then I_1 else I_2 (cond)

$\mid I_1 + I_2$ (sum)

$\mid I_1 \mid I_2$ (par)

$\mid \text{rec } X.\ I$ (rec)

$\mid X$ (recVar)

$\mid (\nu s)\ I$ (res)

Global Calculus

I ::=	$A \rightarrow B : ch(s).\ I$	(init)
|	$A \rightarrow B : s\langle op, e, y\rangle.\ I$	(com)
|	if e@A then I_1 else I_2	(cond)
|	$I_1 + I_2$	(sum)
|	$I_1 \mid I_2$	(par)
|	rec X. I	(rec)
|	X	(recVar)
|	$(\nu s)\ I$	(res)

Global Calculus

I ::=	$A \rightarrow B$: ch(s). I	(init)
|	$A \rightarrow B$: s<op, e,y>. I	(com)
|	if e@A then I_1 else I_2	(cond)
|	$I_1 + I_2$	(sum)
|	$I_1 \mid I_2$	(par)
|	rec X. I	(rec)
|	X	(recVar)
|	(νs) I	(res)

Global Calculus

I ::=	$A \rightarrow B : ch(s).\ I$	(init)
|	$A \rightarrow B : s\langle op, e, y\rangle.\ I$	(com)
|	if $e@A$ then I_1 else I_2	(cond)
|	$I_1 + I_2$	(sum)
|	$I_1 \mid I_2$	(par)
|	rec $X.\ I$	(rec)
|	X	(recVar)
|	$(\nu s)\ I$	(res)

(causality) semantics

$$I \longrightarrow I'$$

- communication happens in the system;
- the system becomes I’

$$A \rightarrow B : ch(s).\ I \quad \rightarrow (\nu s)\ I$$

$$A \rightarrow B : s\langle op, e, y\rangle.\ I \quad \rightarrow I$$

(causality) semantics

$$I \longrightarrow I'$$

- communication happens in the system;
- the system becomes I'

$$A \to B : ch(s).\ I \quad \to (\nu s)\ I$$

$$A \to B : s\langle op, e, y\rangle.\ I \quad \to I[e/y]$$

IT University of Copenhagen

(causality) semantics

$$(\sigma, I) \quad \rightarrow \quad (\sigma', I')$$

- communication happens in the system;
- the system becomes I'

$$(\sigma,\ A \rightarrow B : ch(s).\ I) \rightarrow (\sigma,\ (\nu s)\ I)$$

$$(\sigma,\ A \rightarrow B : s<op,\ e,y>.\ I) \rightarrow (\sigma@B[y = \sigma(e)@A],\ I)$$

An Example

$$
\begin{aligned}
&\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s) \,.\\
&\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathrm{QuoteReq},\ \mathit{product},\ x \rangle \,.\\
&\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathrm{QuoteRes},\ \mathit{quote},\ y \rangle \,.\\
&\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathrm{QuoteAcc},\ \mathit{creditcard},\ z \rangle \,.\\
&\textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r) \,.\\
&\textbf{Seller} \rightarrow \textbf{Shipper} : r\langle \mathrm{ShipReq},\ z,\ x \rangle \,.\\
&\textbf{Shipper} \rightarrow \textbf{Seller} : r\langle \mathrm{ShipConf} \rangle \,.\\
&\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathrm{OrderConf} \rangle \,.\, \mathbf{0}
\end{aligned}
$$

An Example (2)

$\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s)\,.$

$\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteReq},\ \mathit{product},\ x \rangle\,.$

$\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \text{QuoteRes},\ \mathit{quote},\ y \rangle\,.$

$\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteAcc},\ \mathit{creditcard},\ z \rangle\,.$

$\textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r)\,.$

$\cdots \text{(as before)} \cdots$

$+$

$\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteNoGood} \rangle\,.\,\mathbf{0}$

IT University of Copenhagen

An Example (3)

$$\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s) \,.$$
$$\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteReq}, \ \textit{product}, \ x \rangle \,.$$
$$\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \text{QuoteRes}, \ \textit{quote}, \ y \rangle \,.$$

if reasonable(y)@*Buyer* **then**

$$\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteAcc}, \ \textit{creditcard}, \ z \rangle \,.$$
$$\textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r) \,.$$
$\cdots$ (as before) $\cdots$

else

$$\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteNoGood} \rangle \,.\, \mathbf{0}$$

An Example (4)

$$\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s) .$$
$$\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteReq},\ \mathit{product},\ x\rangle .$$
$$\textbf{rec X} . \textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \text{QuoteRes},\ \mathit{quote},\ y\rangle .$$

$$\textbf{if}\ \text{reasonable}(y)@\mathit{Buyer}\ \textbf{then}$$

$$\quad \textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteAcc},\ \mathit{creditcard},\ z\rangle .$$
$$\quad \textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r) .$$
$$\quad \cdots \text{(as before)} \cdots$$

$$\textbf{else}$$

$$\quad \textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteNoGood}\rangle . \textbf{X}$$

End-Point Calculus

P::=	$*ch\ (s)\ .\ P$	(serv)
|	$ch\ !\ (s)\ .\ P$	(req)
|	$s\ ?\ (\ op_1(x_1).P_1\ + \ldots +\ op_n(x_n).P_n\ )$	(in)
|	$s\ !\ op$<e>. P	(out)
|	$P_1 \oplus P_2$	(oplus)
|	$P_1 \mid P_2$	(par)
|	if e then P_1 else P_2	(cond)
|	rec X. P | X | (νs) P	(other)

IT University of Copenhagen

Sunday, October 4, 2009

End-Point Calculus

P::=	*ch (s) . P	(serv)
	| ch ! (s) . P	(req)
	| s ? ($op_1(x_1).P_1$ + ... + $op_n(x_n).P_n$)	(in)
	| s ! op<e>. P	(out)
	| $P_1 \oplus P_2$	(oplus)
	| P_1 | P_2	(par)
	| if e then P_1 else P_2	(cond)
	| rec X. P | X | (νs) P	(other)

End-Point Calculus

P::=	$*ch\ (s)\ .\ P$	(serv)
|	$ch\ !\ (s)\ .\ P$	(req)
|	$s\ ?\ (\ op_1(x_1).P_1\ +\ \dots\ +\ op_n(x_n).P_n\ )$	(in)
|	$s\ !\ op\langle e\rangle.\ P$	(out)
|	$P_1 \oplus P_2$	(oplus)
|	$P_1 \mid P_2$	(par)
|	if e then P_1 else P_2	(cond)
|	rec X. P | X | (νs) P	(other)

IT University of Copenhagen

Sunday, October 4, 2009

End-Point Calculus

P::= $*ch\ (s)\ .\ P$ (serv)

| $ch\ !\ (s)\ .\ P$ (req)

| $s\ ?\ (\ op_1(x_1).P_1 + \ldots + op_n(x_n).P_n\ )$ (in)

| $s\ !\ op\langle e\rangle.\ P$ (out)

| $P_1 \oplus P_2$ (oplus)

| $P_1 \mid P_2$ (par)

| if e then P_1 else P_2 (cond)

| rec X. P | X | (νs) P (other)

IT University of Copenhagen

Sunday, October 4, 2009

End-Point Calculus

P::=	$*ch\ (s)\ .\ P$	(serv)
|	$ch\ !\ (s)\ .\ P$	(req)
|	$s\ ?\ (\ op_1(x_1).P_1\ +\ \ldots\ +\ op_n(x_n).P_n\ )$	(in)
|	$s\ !\ op\langle e\rangle.\ P$	(out)
|	$P_1 \oplus P_2$	(oplus)
|	$P_1 \mid P_2$	(par)
|	if e then P_1 else P_2	(cond)
|	rec X. P | X | (νs) P	(other)

IT University of Copenhagen

Sunday, October 4, 2009

How does EPP work?

For each session (service) and Seller:

$$
\begin{aligned}
&\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s)\,.\\
&\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteReq},\ \textit{product},\ x\rangle\,.\\
&\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \text{QuoteRes},\ \textit{quote},\ y\rangle\,.\\
&\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \text{QuoteAcc},\ \textit{creditcard},\ z\rangle\,.\\
&\textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r)\,.\\
&\textbf{Seller} \rightarrow \textbf{Shipper} : r\langle \text{ShipReq},\ z,\ x\rangle\,.\\
&\textbf{Shipper} \rightarrow \textbf{Seller} : r\langle \text{ShipConf}\rangle\,.\\
&\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \text{OrderConf}\rangle\,.\,\mathbf{0}
\end{aligned}
$$

How does EPP work?

For each session (service) and Seller:

$$
\begin{array}{l}
\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s) \,. \\
\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathrm{QuoteReq},\ \mathit{product},\ x \rangle \,. \\
\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathrm{QuoteRes},\ \mathit{quote},\ y \rangle \,. \\
\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathrm{QuoteAcc},\ \mathit{creditcard},\ z \rangle \,. \\
\textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r) \,. \\
\textbf{Seller} \rightarrow \textbf{Shipper} : r\langle \mathrm{ShipReq},\ z,\ x \rangle \,. \\
\textbf{Shipper} \rightarrow \textbf{Seller} : r\langle \mathrm{ShipConf} \rangle \,. \\
\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathrm{OrderConf} \rangle \,. \mathbf{0}
\end{array}
$$

How does EPP work?

For each session (service) and Seller:

$$
\begin{array}{l}
\mathbf{Buyer} \longrightarrow \mathbf{Seller} : \mathbf{ch1}(s) \,.\\
\mathbf{Buyer} \longrightarrow \mathbf{Seller} : s\langle \mathrm{QuoteReq},\ \mathit{product},\ x\rangle \,.\\
\mathbf{Seller} \longrightarrow \mathbf{Buyer} : s\langle \mathrm{QuoteRes},\ \mathit{quote},\ y\rangle \,.\\
\mathbf{Buyer} \longrightarrow \mathbf{Seller} : s\langle \mathrm{QuoteAcc},\ \mathit{creditcard},\ z\rangle \,.\\
\mathbf{Seller} \longrightarrow \mathbf{Shipper} : \mathbf{ch2}(r) \,.\\
\mathbf{Seller} \longrightarrow \mathbf{Shipper} : r\langle \mathrm{ShipReq},\ z,\ x\rangle \,.\\
\mathbf{Shipper} \longrightarrow \mathbf{Seller} : r\langle \mathrm{ShipConf}\rangle \,.\\
\mathbf{Seller} \longrightarrow \mathbf{Buyer} : s\langle \mathrm{OrderConf}\rangle \,.\, \mathbf{0}
\end{array}
$$

How does EPP work?

For each session (service) and Seller:

Buyer → Seller : ch1(*s*) .
Buyer → Seller : *s*⟨QuoteRe
Seller → Buyer : *s*⟨QuoteRe
Buyer → Seller : *s*⟨QuoteAc
Seller → Shipper : ch2(*r*) .
Seller → Shipper : *r*⟨ShipRe
Shipper → Seller : *r*⟨ShipCo
Seller → Buyer : *s*⟨OrderCo

<u>SELLER</u>

***ch1(s).**
s ? QuoteReq(x).
s ! QuoteRes<*quote*>.
s ? QuoteAcc(z).
ch2!(r).
r ! ShipReq<z>.
r ? ShipConf.
s ! OrderConf

How does EPP work?

and for Buyer...

$$\begin{array}{l}
\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s) \,. \\
\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathrm{QuoteReq},\ \mathit{product},\ x \rangle \,. \\
\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathrm{QuoteRes},\ \mathit{quote},\ y \rangle \,. \\
\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathrm{QuoteAcc},\ \mathit{creditcard},\ z \rangle \,. \\
\textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r) \,. \\
\textbf{Seller} \rightarrow \textbf{Shipper} : r\langle \mathrm{ShipReq},\ z,\ x \rangle \,. \\
\textbf{Shipper} \rightarrow \textbf{Seller} : r\langle \mathrm{ShipConf} \rangle \,. \\
\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathrm{OrderConf} \rangle \,. \mathbf{0}
\end{array}$$

How does EPP work?

and for Buyer...

$$
\begin{array}{l}
\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s) . \\
\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathsf{QuoteReq},\ product,\ x \rangle . \\
\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathsf{QuoteRes},\ quote,\ y \rangle . \\
\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathsf{QuoteAcc},\ creditcard,\ z \rangle . \\
\textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r) . \\
\textbf{Seller} \rightarrow \textbf{Shipper} : r\langle \mathsf{ShipReq},\ z,\ x \rangle . \\
\textbf{Shipper} \rightarrow \textbf{Seller} : r\langle \mathsf{ShipConf} \rangle . \\
\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathsf{OrderConf} \rangle . \mathbf{0}
\end{array}
$$

How does EPP work?

and for Buyer...

Buyer → Seller : ch1(*s*) .
Buyer → Seller : *s*⟨QuoteRe
Seller → Buyer : *s*⟨QuoteRe
Buyer → Seller : *s*⟨QuoteAc
Seller → Shipper : ch2(*r*) .
Seller → Shipper : *r*⟨ShipRe
Shipper → Seller : *r*⟨ShipCo
Seller → Buyer : *s*⟨OrderCo

BUYER

ch1!(s).
s!QuoteReq<*product*>.
s?QuoteRes(y).
s!QuoteAcc<*creditcard*>.
s?OrderConf

How does EPP work? (2)

- Careful with participant with multiple services

$$\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s).$$
$$\textbf{Seller} \rightarrow \textbf{Buyer} : s\langle \mathsf{op}_1,\ e_1,\ x\rangle.$$
$$\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch2}(t).$$
$$\textbf{Seller} \rightarrow \textbf{Buyer} : t\langle \mathsf{op}_2,\ e_2,\ y\rangle.$$
$$\ldots$$
$$\textbf{Buyer2} \rightarrow \textbf{Seller} : \textbf{ch2}(t).$$
$$\textbf{Seller} \rightarrow \textbf{Buyer2} : t\langle \mathsf{op}_3,\ e_3,\ y\rangle.$$

How does EPP work? (2)

- Careful with participant with multiple services

$\mathsf{Buyer} \to \mathsf{Seller} : \mathsf{ch1}(s)\,.$
$\mathsf{Seller} \to \mathsf{Buyer} : s\langle \mathsf{op}_1,\ e_1,\ x\rangle\,.$
$\mathsf{Buyer} \to \mathsf{Seller} : \mathsf{ch2}(t)\,.$
$\mathsf{Seller} \to \mathsf{Buyer} : t\langle \mathsf{op}_2,\ e_2,\ y\rangle\,.$
...
$\mathsf{Buyer2} \to \mathsf{Seller} : \mathsf{ch2}(t)\,.$
$\mathsf{Seller} \to \mathsf{Buyer2} : t\langle \mathsf{op}_3,\ e_3,\ y\rangle\,.$

SELLER

$*\mathbf{ch1(s)}.$
$\mathbf{s\ !\ op_1<e_1>.}$ } *thread*

|

$*\mathbf{ch2!(r)}.$
$\mathbf{r\ !\ op_2<e_2>.}$
$\oplus$
$\mathbf{r\ !\ op_3<e_3>.}$ } *thread*

How does EPP work? (2)

- Careful with participant with multiple services

$\textsf{Buyer} \rightarrow \textsf{Seller} : \textsf{ch1}(s)\,.$
$\textsf{Seller} \rightarrow \textsf{Buyer} : s\langle \textsf{op}_1,\ e_1,\ x\rangle\,.$
$\textsf{Buyer} \rightarrow \textsf{Seller} : \textsf{ch2}(t)\,.$
$\textsf{Seller} \rightarrow \textsf{Buyer} : t\langle \textsf{op}_2,\ e_2,\ y\rangle\,.$
...

SELLER

***ch1(s).**
s ! op$_1$<e$_1$>. } *thread*

e$_2$>. } *thread*

e$_3$>.

NOTE.

The EPP guarantees that actions happen in the right order.

This is not always feasible!!!

IT University of Copenhagen

Sunday, October 4, 2009

Three principles

Three principles for correct EPP:

- **Connectedness**. A causality principle.
- **Well-Threadedness**. A local causality principle related to services.
- **Coherence**. Consistent behaviour of the same service over a global description.

This properties can be (in)validated algorithmically

Three principles

Three principles for correct EPP:

- **Connectedness**. A causality principle.
- **Well-Threadedness**. A local causality principle related to services.
- **Coherence**. Consistent behaviour of the same service over a global description.

This properties can be (in)validated algorithmically

Three principles

Three principles for correct EPP:

- **Connectedness**. A causality principle.
- **Well-Threadedness**. A local causality principle related to services.
- **Coherence**. Consistent behaviour of the same service over a global description.

This properties can be (in)validated algorithmically

Connectedness

Good...	*Bad...*
1. $A \to B : \mathbf{b}(s)$.	a. $A \to B : \mathbf{b}(s)$.
2. $A \to B : s\langle \text{go} \rangle$.	b. $A \to B : s\langle \text{go} \rangle$.
3. $B \to C : \mathbf{c}(t) \ldots$	c. $C \to D : t\langle \text{hello} \rangle \ldots$

- b. is "disconnected" from c.
- we must synchronise either A or B with either C or D
- connectedness: in $A \to B \ldots . C \to D \ldots$ we have $B = C$.

In general, $\{A, B\} \cap \{C, D\} \neq \emptyset$ – not difficult

Coherence

if x@**A then**

$A \rightarrow B : \mathbf{b}(s_1, s_2)$.

$A \rightarrow B : s_1\langle \text{go} \rangle$.

$B \rightarrow A : s_2\langle \text{ok} \rangle . \mathbf{0}$

else

$A \rightarrow B : \mathbf{b}(s_1, s_2)$.

$A \rightarrow B : s_1\langle \text{stop} \rangle$.

$B \rightarrow A : s_2\langle \text{no} \rangle . \mathbf{0}$

if-branch.

$! \mathbf{b}(s_1, s_2) . s_1 \triangleright \text{go} . s_2 \triangleleft \text{ok} . \mathbf{0}$

else-branch.

$! \mathbf{b}(s_1, s_2) . s_1 \triangleright \text{stop} . s_2 \triangleleft \text{no} . \mathbf{0}$

merging the two.

$$! \mathbf{b}(s_1, s_2) . s_1 \triangleright \begin{bmatrix} \text{go} . s_2 \triangleleft \text{ok} . \mathbf{0} + \\ \text{stop} . s_2 \triangleleft \text{no} . \mathbf{0} \end{bmatrix}$$

We project each branch onto **B** and merge the results.

Coherence (bad case)

if x@**A then**

$A \longrightarrow B : \mathbf{b}(s_1, s_2)\,.$

$A \longrightarrow B : s_1\langle \mathrm{go}\rangle\,.$

$B \longrightarrow A : s_2\langle \mathrm{ok}\rangle\,.\,\mathbf{0}$

else

$A \longrightarrow B : \mathbf{b}(s_1, s_2)\,.$

$A \longrightarrow B : s_1\langle \mathrm{go}\rangle\,.$

$B \longrightarrow A : s_2\langle \mathrm{no}\rangle\,.\,\mathbf{0}$

EPP Theorem

The end point projection is:

- Type preserving
- $I \rightarrow I'$ implies $EPP(I) \rightarrow EPP(I')$
- $EPP(I) \rightarrow R$ implies $I \rightarrow I'$ and $EPP(I') \sim R$
- Barb preserving

whenever I is well-typed, connected, well-threaded and coherent

Exceptions

- **Interactional Exceptions and Choreography (08&workinprogress)**
  - Session Types and Exceptions
  - Extension of the Global Calculus
  - About End-Point Projection (EPP)

Exceptions, Literally...

- "An Exception is a person or thing that is excluded from a general statement or does not follow a rule" (Mac Dictionary)
- "Exception (handling) is a programming language construct or computer hardware mechanism designed to handle the occurrence of some condition that changes the normal flow of execution." (Wikipedia)

Exceptions, Literally...

- “An Exception is a person or thing that is excluded from a general statement or does not follow a rule” (Mac Dictionary)

- “Exception (handling) is a programming language construct or computer hardware mechanism designed to handle the occurrence of some condition that changes the normal flow of execution.” (Wikipedia)

IT University of Copenhagen

Exceptions, in General

try { /* Default Code */ }

catch { /* Handler Code */ }

If an exception is thrown by the default code then the handler is executed.

Exceptions are thrown with a special command **throw**

Exceptions and Choreography

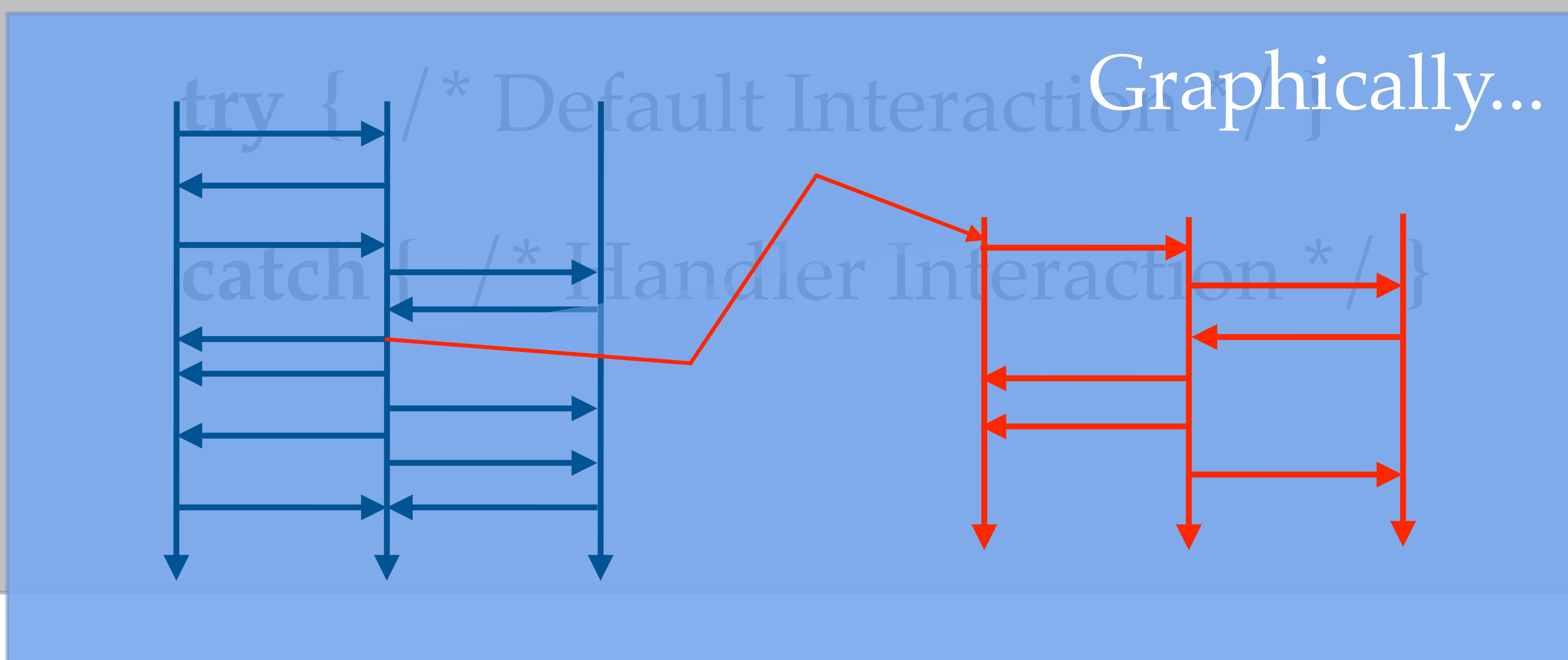
What if apply the exception (compensation) idea to *choreography*?

try { /* Default Interaction */ }

catch { /* Handler Interaction */ }

Exceptions and Choreography

What if apply the exception (compensation) idea to *choreography*?



A Simple Financial Protocol

1. $\texttt{Buyer} \rightarrow \texttt{Seller} \; : \; \mathbf{chSeller}(s) \; [s,$

2. $\mathbf{rec}\ X\ .\ \texttt{Seller} \rightarrow \texttt{Buyer} : s\langle \texttt{update}, \mathit{quote}, y \rangle\ .$
3. $\mathbf{if}\ (y < 100)@\texttt{Buyer}\ \mathbf{then}\ \mathbf{throw}\ \mathbf{else}\ X,$

} default

4. $\texttt{Seller} \rightarrow \texttt{Buyer} : s\langle \texttt{conf}, \mathit{cnum}, x \rangle\ .$
5. $\texttt{Buyer} \rightarrow \texttt{Seller} : s\langle \texttt{data}, \mathit{credit}, x \rangle\ ]$

} handler

A Financial Protocol with Broker

1. $\mathtt{Buyer} \rightarrow \mathtt{Broker} \; : \; \mathbf{chBroker}(s) \; [s,$

2. $\mathtt{Buyer} \rightarrow \mathtt{Broker} : s\langle \mathtt{identify}, id, x\rangle \,.$
3. $\mathbf{if}\ \mathtt{bad}(x)@\mathtt{Broker}\ \mathbf{then\ throw}$
4. $\mathbf{else}\ \mathtt{Broker} \rightarrow \mathtt{Seller} \; : \; \mathbf{chSeller}(t)[(s,t),\ \mathbf{rec}\ X\,.$
5. $\mathtt{Seller} \rightarrow \mathtt{Broker} : t\langle \mathtt{update}, quote, y\rangle \,.$
6. $\mathtt{Broker} \rightarrow \mathtt{Buyer} : s\langle \mathtt{update}, 1.1 * y, y\rangle \,.$
7. $\mathbf{if}\ (y < 100)@\mathtt{Buyer}\ \mathbf{then\ throw\ else}\ X,$

(lines 5–7: default)

8. $\mathtt{Seller} \rightarrow \mathtt{Broker} : t\langle \mathtt{conf}, cnum, x\rangle \,.$
9. $\mathtt{Broker} \rightarrow \mathtt{Buyer} : s\langle \mathtt{conf}, x, x\rangle \,.$
10. $\mathtt{Buyer} \rightarrow \mathtt{Broker} : s\langle \mathtt{data}, credit, x\rangle \,.$
11. $\mathtt{Broker} \rightarrow \mathtt{Seller} : t\langle \mathtt{data}, x, x\rangle],$

(lines 8–11: handler)

(lines 2–11: default)

12. $\mathtt{Broker} \rightarrow \mathtt{Buyer} : s\langle \mathtt{reject}, reason, x\rangle \;]$ (handler)

IT University of Copenhagen

A Financial Protocol with Broker

1. $\texttt{Buyer} \to \texttt{Broker} \ : \ \mathbf{chBroker}(s) \ [s,$
2. $\texttt{Buyer} \to \texttt{Broker} : s\langle \texttt{identify}, id, x\rangle \, .$
3. $\mathbf{if} \ \texttt{bad}(x)@\texttt{Broker} \ \mathbf{then\ throw}$
4. $\mathbf{else} \ \texttt{Broker} \to \texttt{Seller} \ : \ \mathbf{chSeller}(t)[(s,t), \ \mathbf{rec} \ X \, .$
5. $\texttt{Seller} \to \texttt{Broker} : t\langle \texttt{update}, quote, y\rangle \, .$
6. $\texttt{Broker} \to \texttt{Buyer} : s\langle \texttt{update}, 1.1 * y, y\rangle \, .$
7. $\mathbf{if} \ (y < 100)@\texttt{Buyer} \ \mathbf{then\ throw\ else} \ X,$
8. $\texttt{Seller} \to \texttt{Broker} : t\langle \texttt{conf}, cnum, x\rangle \, .$
9. $\texttt{Broker} \to \texttt{Buyer} : s\langle \texttt{conf}, x, x\rangle \, .$
10. $\texttt{Buyer} \to \texttt{Broker} : s\langle \texttt{data}, credit, x\rangle \, .$
11. $\texttt{Broker} \to \texttt{Seller} : t\langle \texttt{data}, x, x\rangle],$
12. ~~$\texttt{Broker} \to \texttt{Buyer} : s\langle \texttt{reject}, reason, x\rangle \ ]$~~

(Lines 5–7: default; lines 8–11: handler; lines 2–11: default; line 12: handler)

Global Calculus Extension

$$
\begin{array}{lllr|lr}
I, J ::= & A \rightarrow B \,\mathbf{:}\, a(s)[\tilde{t}, I, J] & & \text{(init)} & \mid \; \mathbf{0} & \text{(inaction)} \\
\mid & A \rightarrow B : s\langle \mathsf{op}, e, y\rangle \,\mathbf{.}\, I & & \text{(com)} & \mid \; I \mid J & \text{(par)} \\
\mid & \mathbf{throw} & & \text{(throw)} & \mid \; I + J & \text{(sum)} \\
\mid & \mathbf{if}\ e@A\ \mathbf{then}\ I\ \mathbf{else}\ J & & \text{(cond)} & \mid \; X & \text{(recVar)} \\
\mid & \mathbf{rec}\ X \,\mathbf{.}\, I & & \text{(rec)} & &
\end{array}
$$

End-Points?

$P ::=$	$!c(\kappa)[P,Q]$	(service)	$\mid \overline{c}(\lambda)[\tilde{\kappa},P,Q]$	(request)
	$\mid \kappa?\Sigma_i \mathsf{op}_i\langle x_i\rangle.P_i$	(input)	$\mid \kappa!\mathsf{op}\langle e\rangle\,.\,P$	(output)
	$\mid P \mid Q$	(par)	$\mid \textbf{if}\ e\ \textbf{then}\ P\ \textbf{else}\ P$	(cond)
	$\mid \mathbf{0}$	(inact)	$\mid P \oplus Q$	(choice)
	$\mid X$	(termVar)	$\mid \textbf{rec}\ X\,.\,P$	(recursion)
	$\mid \textbf{throw}$	(throw)		

$$N ::= \quad A[\ P\ ]_\sigma \mid \quad N_1 \mid\mid N_2 \mid \quad \epsilon$$

IT University of Copenhagen

Sunday, October 4, 2009

EPP?

IT University of Copenhagen

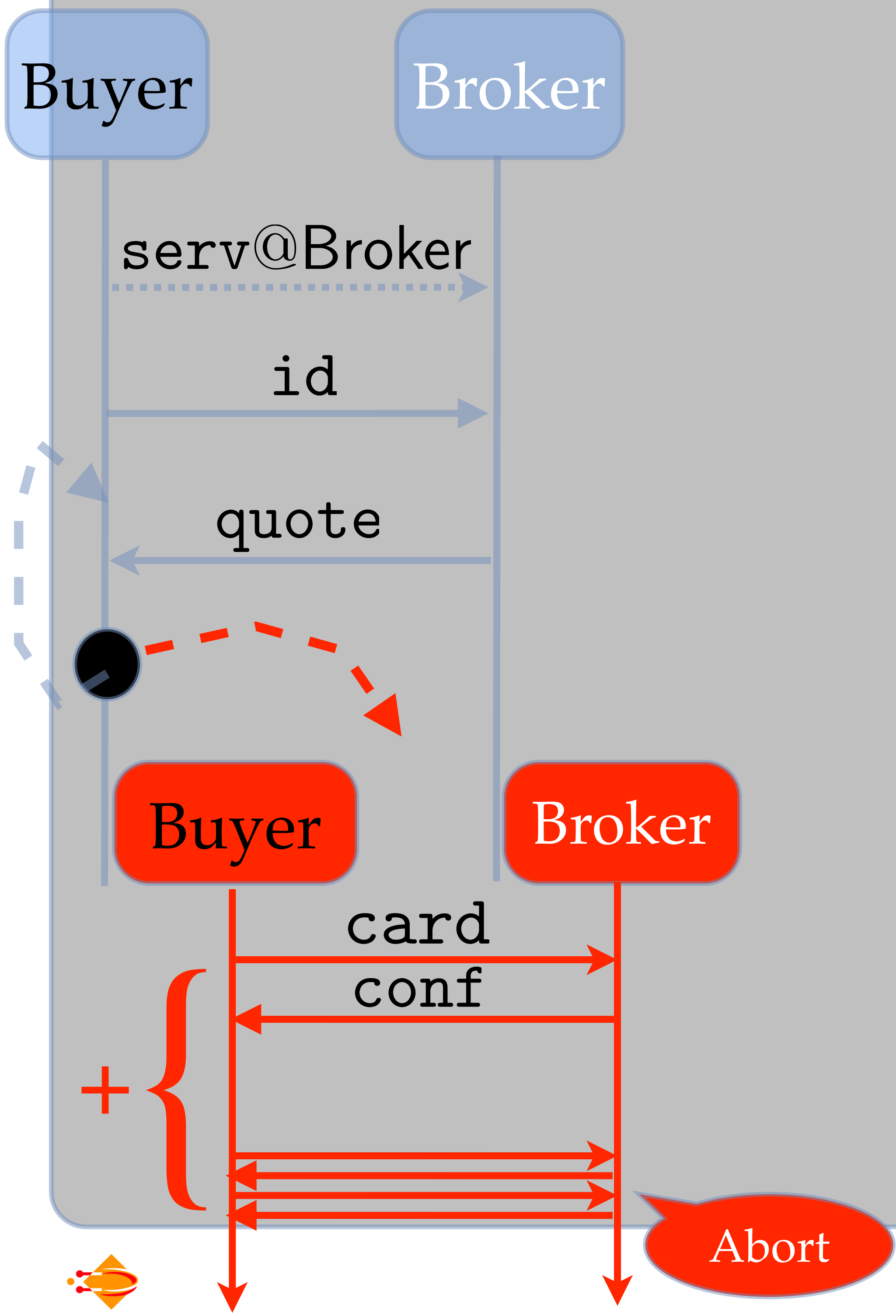
EPP? (Buyer)

$$\overline{\textbf{chBroker}}(s^+)[s^+,$$

$$s^+!\texttt{identify}\langle id\rangle\,.\,\textbf{rec}\ X\,.\,s^+?\texttt{update}(y)\,.\,\textbf{if}\ (y < 100)\ \textbf{then throw else}\ X,$$

$$\mathsf{merge}(s^+?\mathsf{conf}(x)\,.\,s^+!\mathsf{data}\langle credit\rangle,\ s^+?\mathsf{reject}(x)\,.\,P)\ ]$$

EPP? (Buyer)



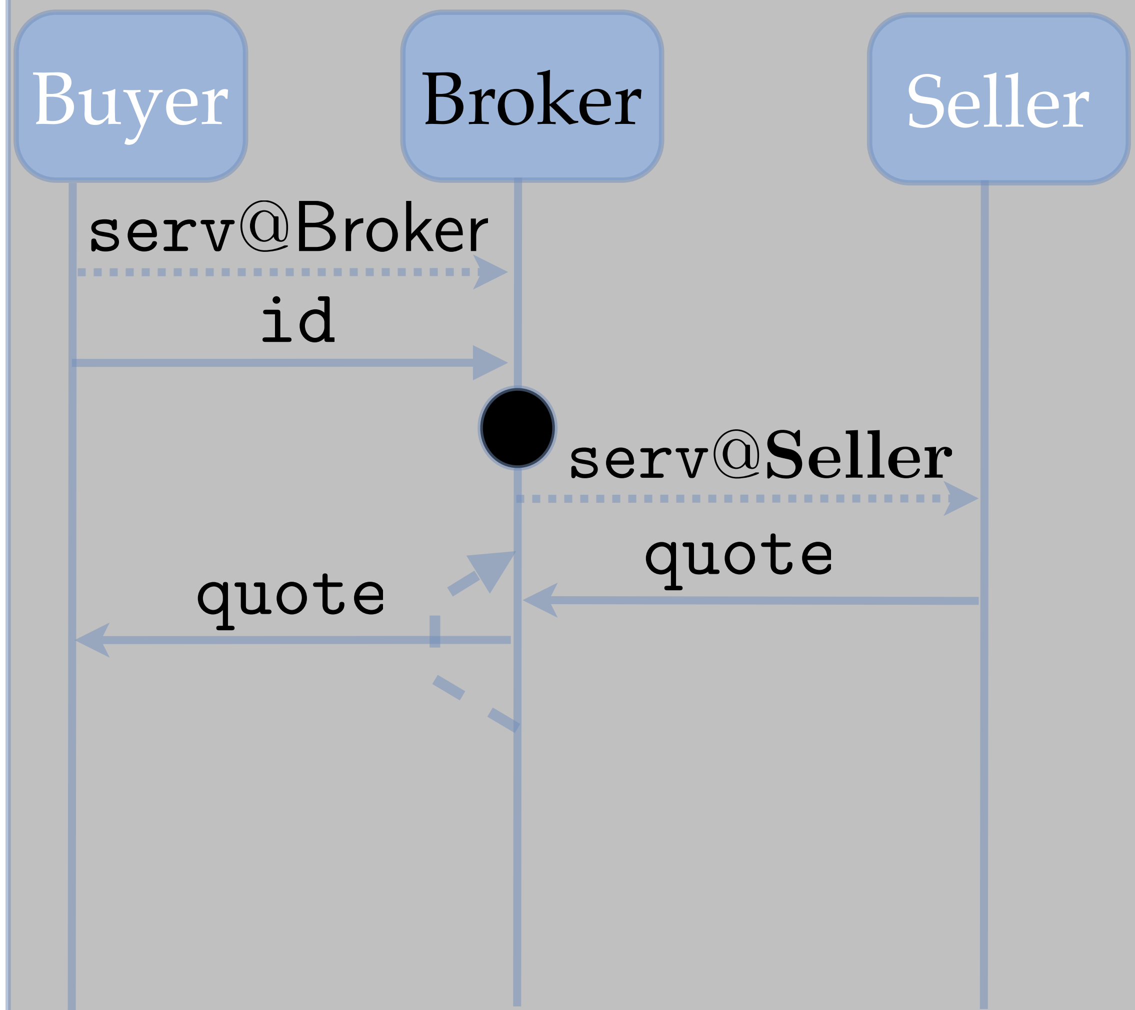
Buyer

$$
\begin{array}{l}
\textbf{invoke}\ \texttt{serv@Broker}(t);\\
\quad \textbf{try}\ \{\\
\qquad t!\langle \texttt{id}\rangle.\\
\qquad \mu X.\\
\qquad t?(y).\\
\qquad \textbf{if}\ (ok(y))\ \textbf{then throw else}\ X\\
\quad \}\\
\quad \textbf{catch}\ \{\\
\qquad l_1\ :\ t!\langle \texttt{card}\rangle.\ t?(y)\\
\qquad l_2\ :\ P_{\texttt{abort}}\\
\quad \}
\end{array}
$$

EPP? (Broker)

$$
\left.
\begin{array}{l}
*\mathbf{chBroker}(s^-)[\\
\quad \left.\begin{array}{l}
s^-?\mathsf{identify}(x)\,.\,\mathbf{if}\ \mathtt{bad}(x)\ \mathbf{then\ throw} \\
\mathbf{else}\ \overline{\mathbf{chSeller}}\,(t^+)[(s^-,t^+), \\
\quad \mathbf{rec}\ X\,.\ t^+?\mathtt{update}(y)\,.\,s^-!\mathtt{update}\langle y+10\%\rangle\,.\,X \quad \}\ \text{default} \\
\quad t^+?\mathtt{conf}(x)\,.\,s^-!\mathtt{conf}\langle x\rangle\,.\,s^-?\mathtt{data}(x)\,.\,t^+!\mathtt{data}\langle x\rangle], \quad \}\ \text{handler}
\end{array}\right\}\ \text{default} \\
\quad s^-!\mathtt{reject}\langle \mathit{reason}\rangle\,.\,\mathcal{A}\ ] \quad \}\ \text{handler}
\end{array}
\right.
$$

EPP? (Broker)

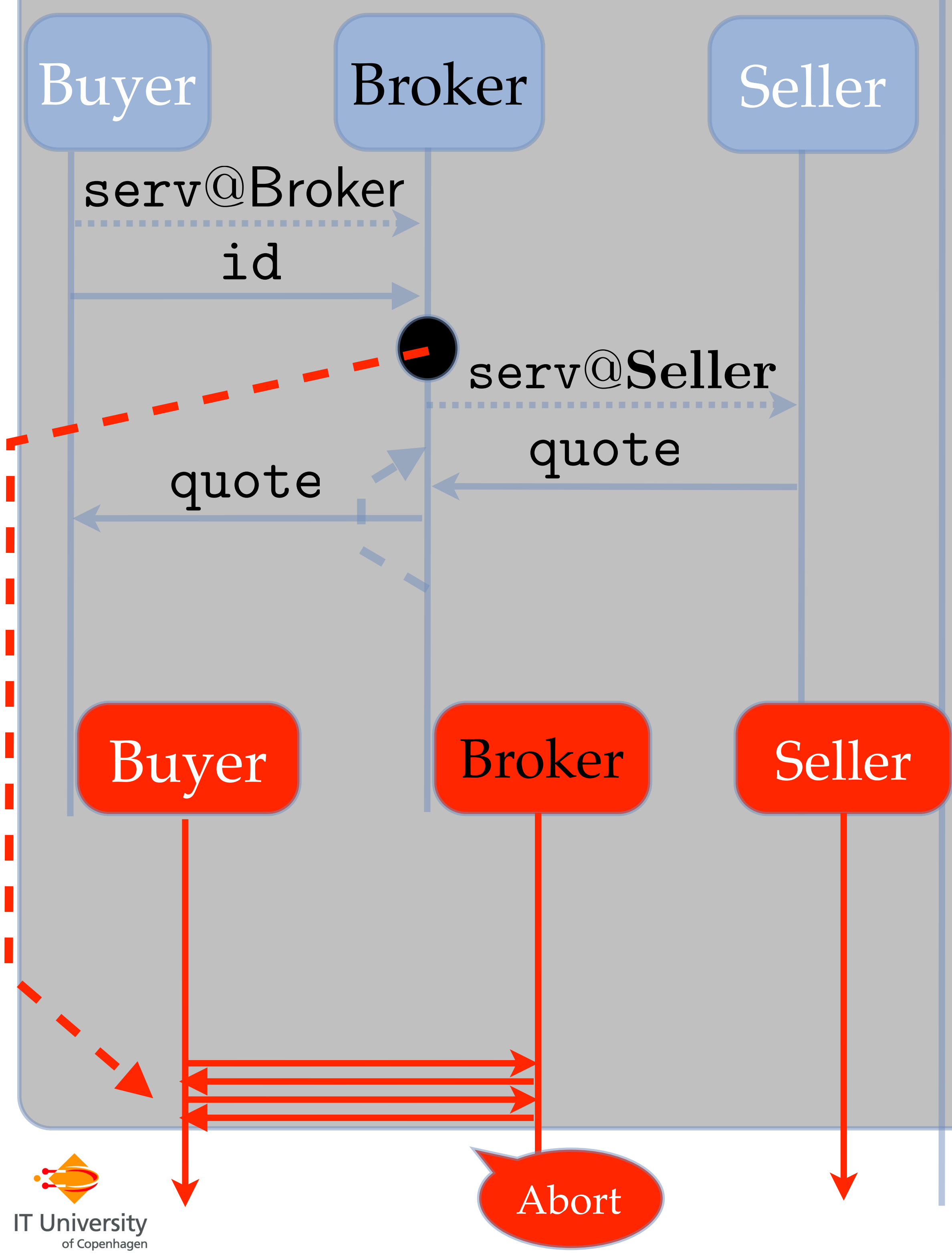


<u>Broker</u>

service serv(t) {
try (t) {
$t?(x)$.
if bad(x) **then throw else**
invoke serv@Seller(s);
try (t, s) {
$\mu X.\ s?(x).\ t!\langle x + 10\%\rangle.\ X$
catch $\{\ t \triangleright l_1.\ \ldots$fwd$\ldots\ \}$

catch $\{\ t \triangleright l_2.\ P'_{\texttt{abort}}\ \}$
}

EPP? (Broker)



Broker

service serv(t) {

try (t) {

$t?(x)$.

if bad(x) **then throw else**

invoke serv@Seller(s);

try (t, s) {

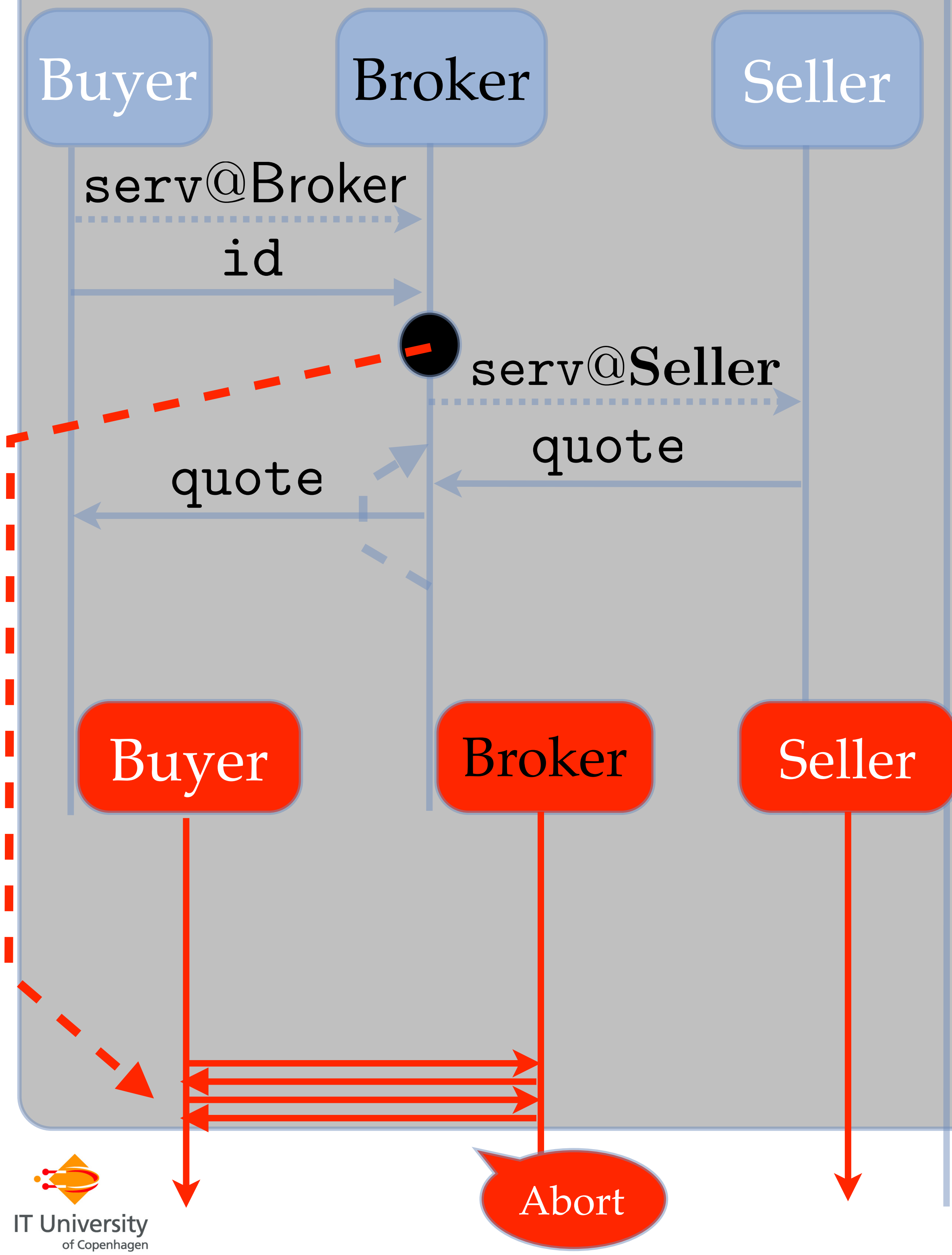
$\mu X.\ s?(x).\ t!\langle x + 10\% \rangle.\ X$

catch $\{\ t \rhd l_1.\ \ldots$fwd$\ldots\ \}$

catch $\{\ t \rhd l_2.\ P'_{\texttt{abort}}\ \}$

}

EPP? (Broker)



Broker

service serv(t) {

try (t) {

$t?(x).$

if bad(x) **then throw else**

invoke serv@Seller(s);

try (t, s) {

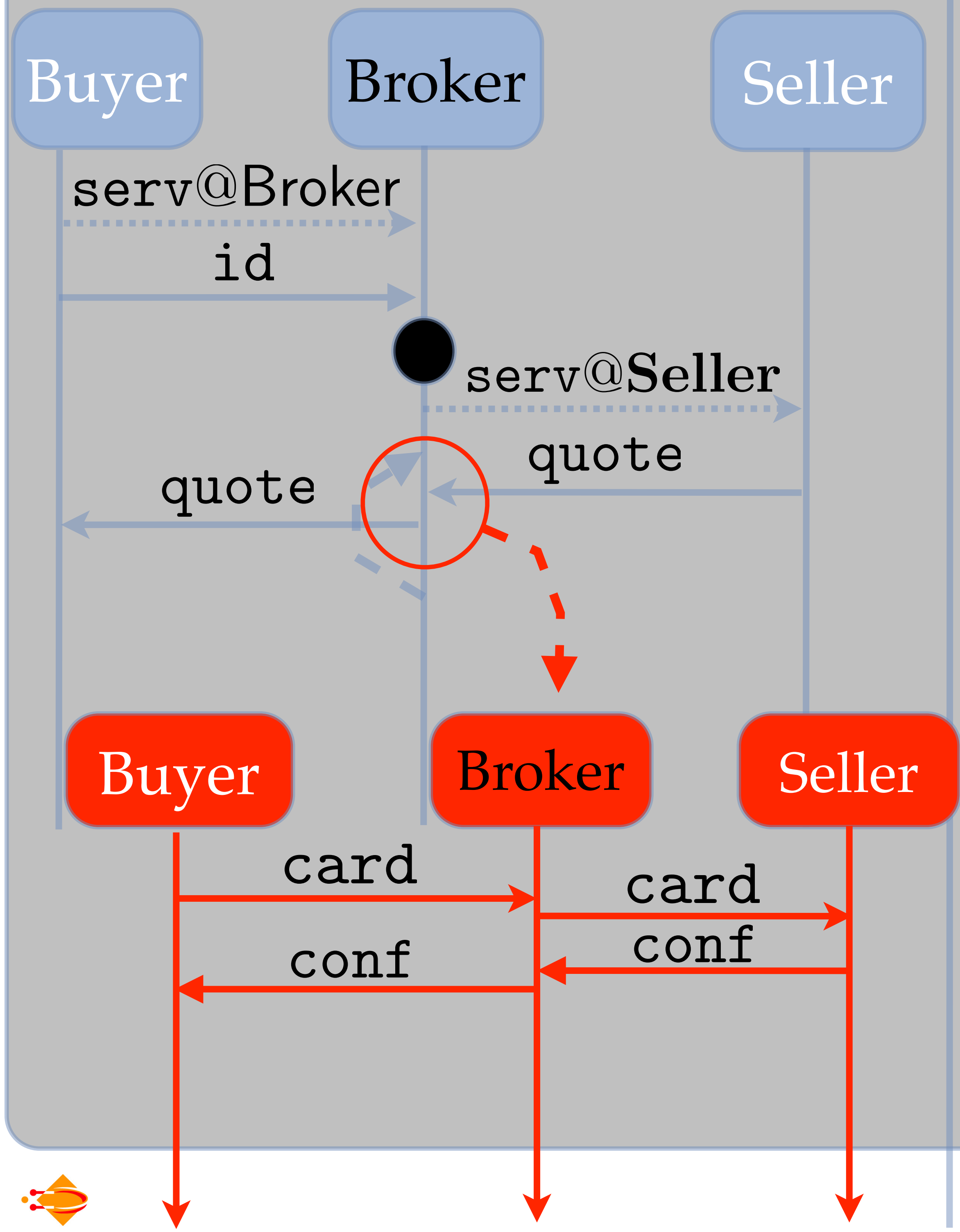
$\mu X.\ s?(x).\ t!\langle x + 10\% \rangle.\ X$

catch $\{\ t \triangleright l_1.\ \ldots \mathrm{fwd} \ldots\ \}$

catch $\{\ t \triangleright l_2.\ P'_{\texttt{abort}}\ \}$

}

EPP? (Broker)



Broker

service serv(t) $\{$

try (t) $\{$

$t?(x).$

if bad(x) **then throw else**

invoke serv@Seller(s);

try (t, s) $\{$

$\mu X.\ s?(x).\ t!\langle x + 10\% \rangle.\ X$

catch $\{\ t \triangleright l_1.\ \ldots \text{fwd} \ldots\ \}$

~~**catch** $\{\ t \triangleright l_2.\ P'_{\texttt{abort}}\ \}$~~

$\}$

EPP? (Seller)

$$*\mathbf{chSeller}(t^-)[\ \mathbf{rec}\ X\,.\,t^-!\mathsf{update}\langle quote\rangle\,.\,X,\ \ t^-!\mathsf{conf}\langle cnum\rangle\,.\,t^-?\mathsf{data}(x)\ ]$$

Other Stuff on Choreography

- Choreography as a Session Type (Multiparty Session Types) [@POPL08] [Yoshida et al. @CONCUR08, ESOP09]
- Choreography and Strand Spaces (Security) [PLACES09, ICE09 - jointly with J.Guttman]
- Scribble, a language based on choreography and session types (Pi4Tech, K. Honda, R. Hu)
- ...

IT University of Copenhagen

Work in Progress

- **Logic** for choreography and Partial Specification
- **Annotated** Multiparty Session Types
- **Secure EPP** for Choreography

- Global View Extraction (Inverse of EPP)

An open question?

An open question?

- How do you do it? Still an open problem (in general)...
- Allows for **round-trip engineering** (reuse&de-coupling)
    - code ⇒ choreography (change) ⇒ code

Thank you

Well-Threadedness

$A \rightarrow B : \mathbf{b}(s) .$
$A \rightarrow B : s\langle \mathrm{go} \rangle .$
$B \rightarrow C : \mathbf{c}(t) .$
$C \rightarrow A : \mathbf{a}(r) .$
$A \rightarrow C : r\langle \mathrm{hi} \rangle .$
$C \rightarrow B : t\langle \mathrm{acc} \rangle .$
$B \rightarrow A : s\langle \mathrm{ok} \rangle .$
$\ldots$

$$\mathbf{A} \left[\begin{array}{l} \overline{\mathbf{b}}(\nu s) . s \lhd \mathrm{go} . s \rhd \mathrm{ok} . \ldots \mid \\ !\mathbf{a}(r) . r \lhd \mathrm{hi} . \ldots \end{array} \right]$$

$$\mathbf{B} \left[\begin{array}{l} !\mathbf{b}(s) . s \rhd \mathrm{go} . \\ \quad \overline{\mathbf{c}}(\nu t) . t \rhd \mathrm{acc} . s \lhd \mathrm{ok} . \ldots \end{array} \right]$$

$$\mathbf{C} \left[!\mathbf{c}(t) . \overline{\mathbf{a}}(\nu r) . r \rhd \mathrm{hi} . t \lhd \mathrm{acc} . \ldots \right]$$

Well-Threadedness (2)

Bad...

$\mathsf{A} \rightarrow \mathsf{B} : \mathbf{b}(s).$
$\mathsf{A} \rightarrow \mathsf{B} : s\langle \mathrm{go} \rangle.$
$\mathsf{B} \rightarrow \mathsf{C} : \mathbf{c}(t).$
$\mathsf{C} \rightarrow \mathsf{A} : \mathbf{a}(r).$
$\mathsf{A} \rightarrow \mathsf{C} : r\langle \mathrm{hi} \rangle.$
$\mathsf{C} \rightarrow \mathsf{A} : r\langle \mathrm{hi} \rangle.$
$\mathsf{A} \rightarrow \mathsf{B} : s\langle \mathrm{hello} \rangle.$
$\ldots$

Good...

$\mathsf{A} \rightarrow \mathsf{B} : \mathbf{b}(s).$
$\mathsf{A} \rightarrow \mathsf{B} : s\langle \mathrm{go} \rangle.$
$\mathsf{B} \rightarrow \mathsf{C} : \mathbf{c}(t).$
$\mathsf{C} \rightarrow \mathsf{A} : \mathbf{a}(r).$
$\mathsf{A} \rightarrow \mathsf{C} : r\langle \mathrm{hi} \rangle.$
$\mathsf{C} \rightarrow \mathsf{B} : t\langle \mathrm{acc} \rangle.$
$\mathsf{B} \rightarrow \mathsf{A} : s\langle \mathrm{ok} \rangle.$
$\ldots$

Well-Threadedness (3)

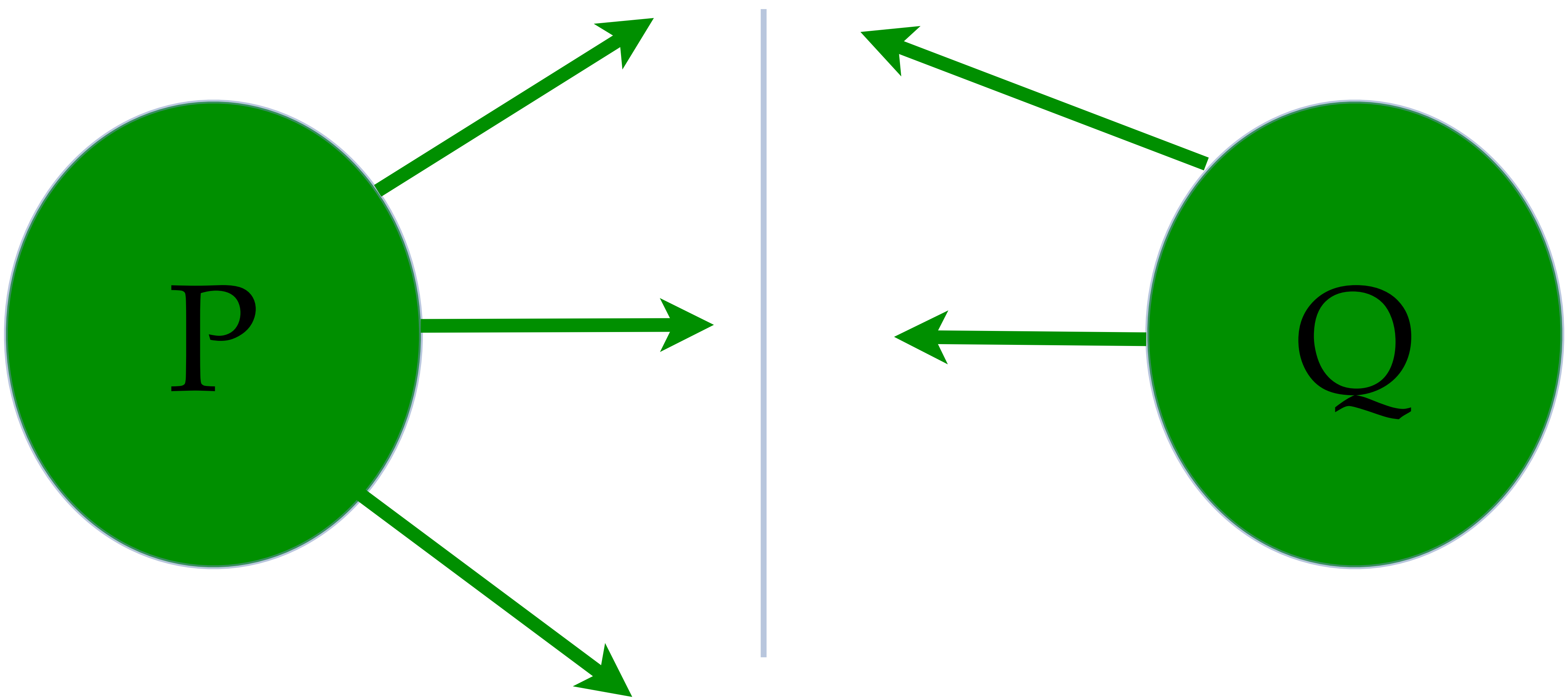
$$\begin{array}{l}
A \longrightarrow B : \mathbf{b}(s). \\
A \longrightarrow B : s\langle \mathrm{go} \rangle. \\
B \longrightarrow C : \mathbf{c}(t). \\
C \longrightarrow A : \mathbf{a}(r). \\
A \longrightarrow C : r\langle \mathrm{hi} \rangle. \\
C \longrightarrow A : r\langle \mathrm{hi} \rangle. \\
A \longrightarrow B : s\langle \mathrm{hello} \rangle. \\
\dots
\end{array}$$

$$\mathbf{A} \left[\begin{array}{l} \overline{\mathbf{b}}(s).\ s \triangleleft \mathrm{go}.\ \underline{s \triangleleft \mathrm{hello}}.P \\ !\mathbf{a}(r).\ r \triangleleft \mathrm{hi}.\ r \triangleright \mathrm{hi}.P' \end{array} \right]$$

$$\mathbf{B} \left[\begin{array}{l} !\mathbf{b}(s).\ s \triangleright \mathrm{go}. \\ \qquad \overline{\mathbf{c}}(t).\ \underline{s \triangleright \mathrm{hello}}.Q \end{array} \right]$$

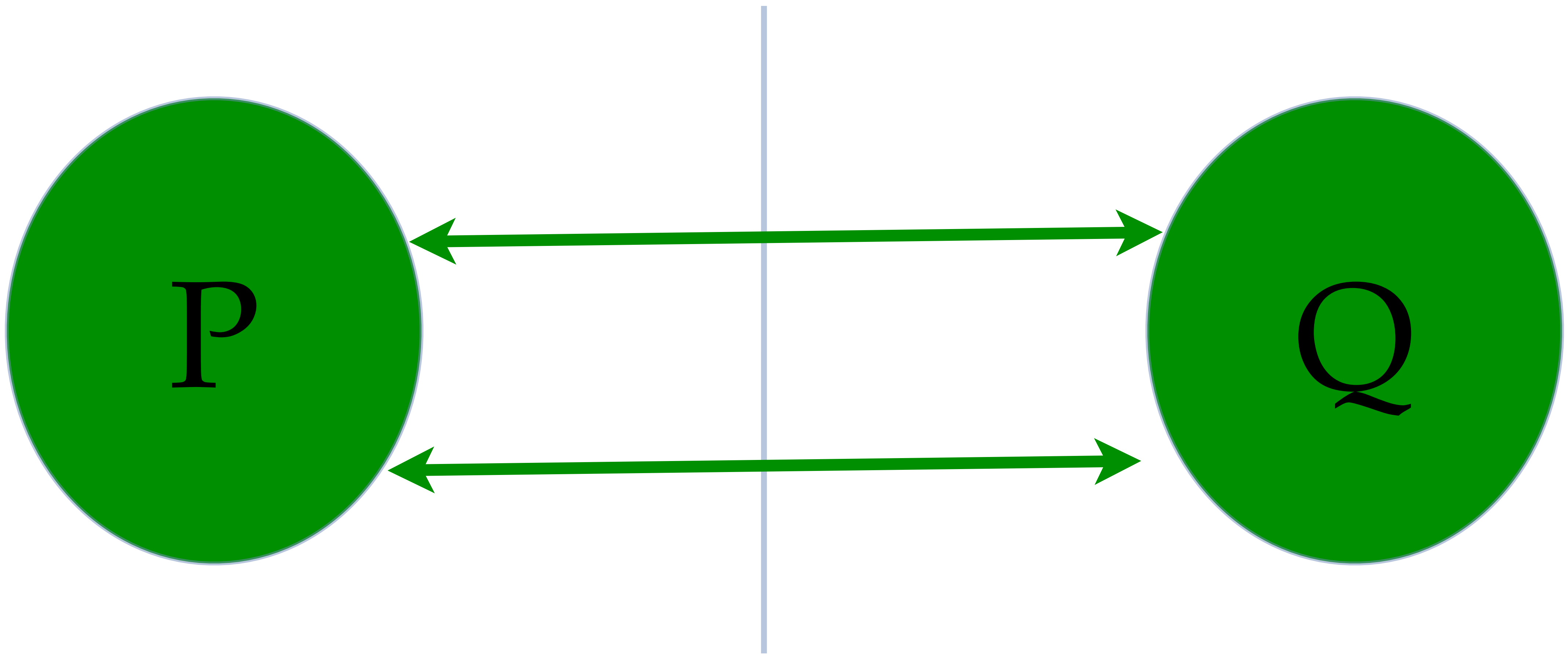
$$\mathbf{C} \left[!\mathbf{c}(t).\ \overline{\mathbf{a}}(r).\ r \triangleright \mathrm{hi}.\ r \triangleleft \mathrm{hi}.S \right]$$

Parallel



IT University of Copenhagen

Parallel



Parallel