A Theoretical Basis of Communication-Centred Programming for *Web Service*

Nobuko Yoshida (Imperial)
Kohei Honda (Queen Mary)

TiC'2006, July 2006

In Collaboration with:

Marco Carbone (Queen Mary)

Vasco Vasconcelos (Lisbon)

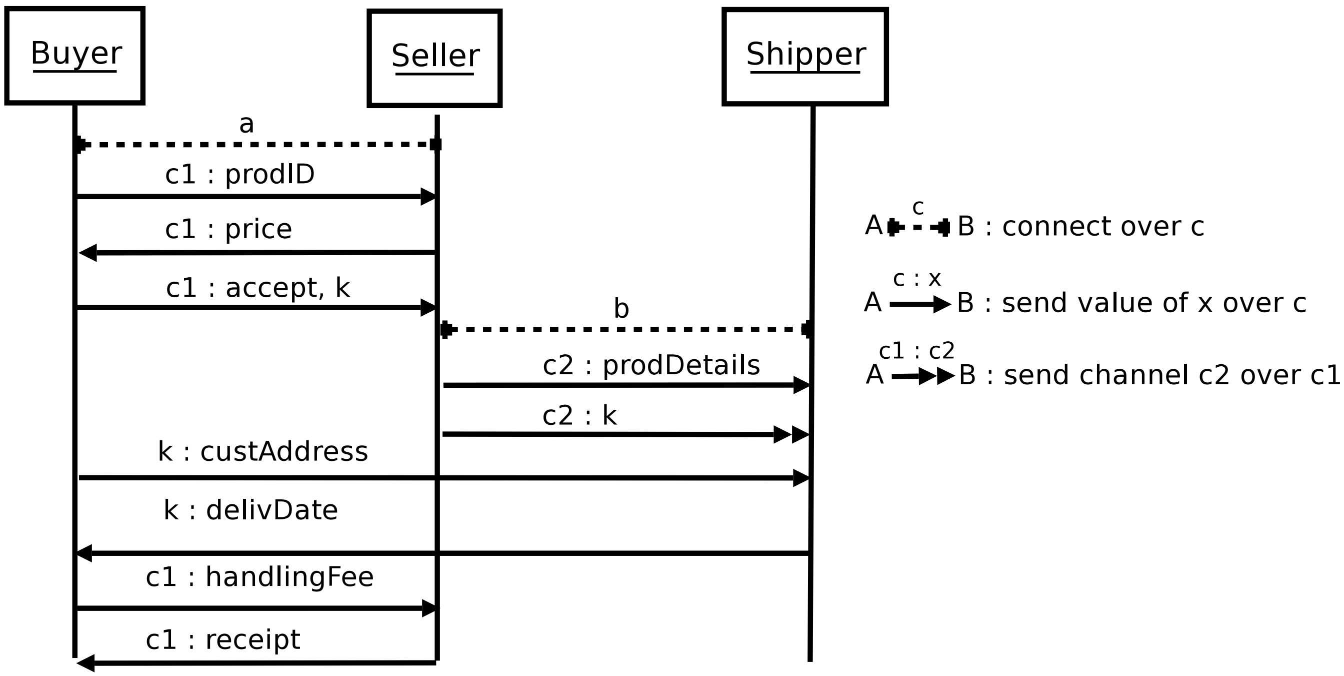
Gary Brown (pi4 technologies)

Steve Ross-Talbot (pi4 technologies)

Structure of Lectures

- **Part 1** Basic Theory (Mobile Processes and Types)
  - 1 Introduction to the π-Calculus
  - 2 Idioms for Interactions
  - 3 Session Types
- **Part 2** Web Services and the π-Calculus
  - 1 Web Services Choreography Description Language
  - 2 Global Language and the End-Point Calculus
  - 3 End-Point Projection and Correctness

Protocol Example



Scenario: Item Purchasing (Typical W3C example)

Challenges

- How can we design languages for Web Services?

  $\Longrightarrow$ use the π-calculus as an underlying formal model

- What are good programming and type disciplines for Web Services?

  $\Longrightarrow$ use the type theory of the π-calculus (session types) for structured programming of communication and concurrency

- How can we validate correctness of Web Services?

  $\Longrightarrow$ use a semantics, type and structured preserving translation from Web Service languages to the π-calculus

Syntax

- Names: $a, b, c, \ldots, x, y, z, \ldots$.
- the Asynchronous π-Calculus

  (Honda and Tokoro 1991, Boudol 1992)

$$P ::= \mathbf{0} \mid a(x).P \mid \overline{a}\langle b\rangle \mid P|Q \mid (\nu x)P \mid !a(x).P$$

- cf. CCS

$$P ::= \mathbf{0} \mid a(x).P \mid \overline{a}(b).\mathbf{0} \mid P|Q \mid P\backslash\{x\} \mid A \stackrel{\text{def}}{=} P$$

Computation

- **CCS** Interaction = Synchronisation

$$(a.P + R) \mid (\overline{a}.R + Q) \longrightarrow P \mid R$$

- **π** Interaction = (Synchronisation and) Name-Passing

$$a(x).P \mid \overline{a}\langle b\rangle \longrightarrow P\{b/x\}$$

  - Internal choice: $P \oplus Q \;=\; (\nu c)(\overline{c} \mid c.P \mid c.Q)$

Binding

- **Association** $|$ is the weakest.
  - $(\nu x)a(y).P = ((\nu x)(a(y).P))$ and $(\nu x)P \mid Q = ((\nu x)P) \mid Q$
  - $(\nu y)a(x).P = (\nu y)(a(x).P)$, $(\nu y)!a(x).P = (\nu y)(!a(x).P)$.
- **Free Names** $\mathrm{fn}(P)$
  - $a(x).\overline{b}\langle x\rangle \qquad a(x).x(z).\mathbf{0}$
  - $(\nu a)a(x).\overline{x}\langle v\rangle$
  - $(\nu a)a(x).\overline{x}\langle v\rangle \mid b(x).\overline{a}\langle x\rangle$

Structure Congruence

- To handle the parts of terms with no computational significance
- Inspired by Chemical Abstract Machine (Berry and Boudol 1991)
- $P \equiv Q$
  - Change of bound names (α-conversion).
  - $P|\mathbf{0} \equiv P \qquad P|Q \equiv Q|P \qquad (P|Q)|R \equiv P|(Q|R)$
  - $(\nu x)\mathbf{0} \equiv \mathbf{0} \qquad (\nu xx)P \equiv (\nu x)P$
    $(\nu xy)P \equiv (\nu yx)P$
  - $((\nu x)P)|Q \equiv (\nu x)(P|Q) \qquad (x \notin \mathrm{fn}(Q))$

Examples (1)

- $\mathbf{0} \mid \mathbf{0} \mid \mathbf{0} \equiv \mathbf{0}$.
- $(\nu a)(\overline{a}\langle v\rangle \mid \mathbf{0}) \equiv (\nu a)\overline{a}\langle v\rangle$.
- $(\nu a)(\overline{b}\langle v\rangle \mid \mathbf{0}) \equiv \overline{b}\langle v\rangle \mid (\nu a)\mathbf{0} \equiv \overline{b}\langle v\rangle$.
- $(\nu z)(\overline{x}\langle z\rangle \mid z(w).\overline{c}\langle w\rangle) \mid x(y).\overline{z}\langle y\rangle$
  $\equiv (\nu z')(\overline{x}\langle z'\rangle \mid z'(w).\overline{c}\langle w\rangle \mid x(y).\overline{z}\langle y\rangle)$

Reduction Relation

Com $\quad x(y).P \mid \overline{x}\langle v\rangle \longrightarrow P\{v/y\}$

Rep $\quad !x(y).P \mid \overline{x}\langle v\rangle \longrightarrow P\{v/y\} \mid !x(y).P$

Par $\quad \dfrac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$ $\qquad$ Res $\quad \dfrac{P \longrightarrow P'}{(\nu x)P \longrightarrow (\nu x)P'}$

Struct $\quad \dfrac{Q \equiv P \quad P \longrightarrow P' \quad P' \equiv Q'}{Q \longrightarrow Q'}$

Examples (1): Forwarder

Let $\mathtt{FW}(ab) = !a(x).\overline{b}\langle x\rangle$. Then

$$\mathtt{FW}(ab) \mid \overline{a}\langle v\rangle \longrightarrow \overline{b}\langle v\rangle \mid \mathtt{FW}(ab).$$

- $$\mathtt{FW}(ab) \mid \overline{a}\langle v\rangle \mid \overline{a}\langle w\rangle \longrightarrow \mathtt{FW}(ab) \mid \overline{b}\langle v\rangle \mid \overline{a}\langle w\rangle \longrightarrow \mathtt{FW}(ab) \mid \overline{b}\langle v\rangle \mid \overline{b}\langle w\rangle$$

  We also have:

  $$\mathtt{FW}(ab) \mid \overline{a}\langle v\rangle \mid \overline{a}\langle w\rangle \longrightarrow \mathtt{FW}(ab) \mid \overline{a}\langle v\rangle \mid \overline{b}\langle w\rangle \longrightarrow \mathtt{FW}(ab) \mid \overline{b}\langle v\rangle \mid \overline{b}\langle w\rangle$$

- $$\overline{a}\langle v\rangle \mid \mathtt{FW}(ab) \mid \mathtt{FW}(bc) \longrightarrow \mathtt{FW}(ab) \mid \overline{b}\langle v\rangle \mid \mathtt{FW}(bc) \longrightarrow \mathtt{FW}(ab) \mid \mathtt{FW}(bc) \mid \overline{c}\langle v\rangle.$$

Scope Opening

- $(\nu x)(\overline{a}\langle x\rangle \mid x(y).\overline{d}\langle y\rangle) \mid a(z).\overline{z}\langle w\rangle$

  $\equiv (\nu x)(\overline{a}\langle x\rangle \mid x(y).\overline{d}\langle y\rangle \mid a(z).\overline{z}\langle w\rangle)$

  $\equiv (\nu x)(x(y).\overline{d}\langle y\rangle \mid \overline{a}\langle x\rangle \mid a(z).\overline{z}\langle w\rangle)$

  $\longrightarrow (\nu x)(x(y).\overline{d}\langle y\rangle \mid \overline{x}\langle w\rangle)$

  $\longrightarrow (\nu x)\overline{d}\langle w\rangle \equiv \overline{d}\langle w\rangle.$

Exercise (1)

1. $\overline{a}\langle v\rangle \mid \overline{b}\langle w\rangle \mid \mathtt{FW}(ab) \mid \mathtt{FW}(bc)$
2. $\overline{a}\langle v\rangle \mid \overline{b}\langle w\rangle \mid (\nu\, b')(\mathtt{FW}(ab') \mid \mathtt{FW}(b'c))$
3. $(\nu\, x)(\overline{a}\langle x\rangle \mid x(y).\overline{d}\langle y\rangle) \mid a(z).\overline{z}\langle w\rangle \mid a(z).\overline{z}\langle v\rangle$
4. $\overline{a}\langle x\rangle \mid x(y).\overline{d}\langle y\rangle \mid a(z).\overline{z}\langle w\rangle \mid x(z).\overline{z}\langle v\rangle$
5. $(\nu\, x)(\overline{a}\langle x\rangle \mid !x(y).\overline{d}\langle y\rangle) \mid a(z).(\overline{z}\langle w\rangle \mid \overline{z}\langle w'\rangle)$

Small Agents (1)

- **New Name Creator** $\mathtt{N}(a) \stackrel{\text{def}}{=} (\nu x)!a(y).\overline{y}\langle x\rangle$

  $\mathtt{N}(a) \mid \overline{a}\langle b\rangle \mid \overline{a}\langle c\rangle \longrightarrow\longrightarrow \mathtt{N}(a) \mid (\nu x)\overline{b}\langle x\rangle \mid (\nu x)\overline{c}\langle x\rangle$

- **Identity Receptor** $\mathtt{FW}(aa)$

  $\mathtt{FW}(aa) \mid \overline{a}\langle v\rangle \longrightarrow \mathtt{FW}(aa) \mid \overline{a}\langle v\rangle$

- **Equator** $\mathtt{EQ}(ab) \stackrel{\text{def}}{=} (\mathtt{FW}(ab) \mid \mathtt{FW}(ba)).$

  Note that $\mathtt{EQ}(ab) \equiv \mathtt{EQ}(ba)$.

  $\mathtt{EQ}(ab) \mid \overline{a}\langle v\rangle$

  $\mathtt{EQ}(ab) \mid \overline{c}\langle a\rangle \cong \mathtt{EQ}(ab) \mid \overline{c}\langle b\rangle$

Small Agents (2)

- **Distributor** $\mathtt{D}(abc) \stackrel{\text{def}}{=} a(x).(\overline{b}\langle x\rangle \mid \overline{c}\langle x\rangle)$

  $\mathtt{D}(abcd) \stackrel{\text{def}}{=} (\nu\, c_1)(\mathtt{D}(abc_1) \mid \mathtt{D}(c_1cd))$

  - $a(x).(P \mid Q) = (\nu\, c_1c_2)(\mathtt{D}(ac_1c_2) \mid c_1(x).P \mid c_2(x).Q)$

- **Killer** $\mathtt{K}(a) \stackrel{\text{def}}{=} a(x).\mathbf{0}$
- **Left Binder** $\mathtt{Br}(ab) \stackrel{\text{def}}{=} a(x).\mathtt{FW}(xb)$
- **Right Binder** $\mathtt{Bl}(ab) \stackrel{\text{def}}{=} a(x).\mathtt{FW}(bx)$
- **Synchroniser** $\mathtt{S}(abc) \stackrel{\text{def}}{=} a(x).\mathtt{FW}(bc)$

Joyful *Hacking* in the π-Calculus

Synchrony in Asynchrony

- Synchronous π-Calculus

$$P \ ::= \ \mathbf{0} \ | \ a(x).P \ | \ \overline{a}\langle b\rangle.P \ | \ P|Q \ | \ (\nu x)P \ | \ !a(x).P$$

- Reduction $x(y).P \,|\, \overline{x}\langle v\rangle.Q \longrightarrow P\{v/y\} \,|\, Q.$

- Mapping $(\ )^\star$: Synchronous π → Asynchronous π

$$(x(y).P)^\star = (\nu c)(\overline{x}\langle c\rangle \,|\, c(y).P^\star)$$
$$(\overline{x}\langle v\rangle.P)^\star = x(y).(\overline{y}\langle v\rangle \,|\, P^\star)$$

Polyadicity in Mondadicity

- Polyadic π-Calculus ($n \geq 0$)

$$P \;::=\; a(x_1,x_2,\ldots,x_n).P \mid \overline{a}\langle b_1,b_2,\ldots,b_n\rangle.P$$
$$\mid \quad !a(x_1,x_2,\ldots,x_n).P \mid \cdots$$

- $x(y_1,y_2,\ldots,y_n).P \mid \overline{x}\langle v_1,v_2,\ldots,v_n\rangle.Q$

$$\longrightarrow P\{v_1/y_1\}\{v_2/y_2\}\ldots\{v_n/y_n\} \mid Q.$$

- We can use the macro $\overline{a}(c).P$ means $(\nu c)\overline{a}\langle c\rangle.P$
- Mapping $(\ )^*$: Polyadic π → Synchronous π

$$(x(y_1,y_2,\ldots,y_n).P)^* = x(c).c(y_1).c(y_2)\ldots c(y_n).P^*.$$
$$(\overline{x}\langle v_1,v_2,\ldots,v_n\rangle.P)^* = \overline{x}(c).\overline{c}\langle v_1\rangle.\overline{c}\langle v_2\rangle\ldots\overline{c}\langle v_n\rangle.P^*.$$

Exercises

- Why the following mapping is incorrect?

  $(x(y_1, y_2, \ldots, y_n).P)^* = x(y_1).x(y_2)\ldots x(y_n).P^*.$

  $(\overline{x}\langle v_1, v_2, \ldots, v_n\rangle.P)^* = \overline{x}\langle v_1\rangle.\overline{x}\langle v_2\rangle\ldots\overline{x}\langle v_n\rangle.P^*.$

- Sequencing

  $a(\tilde{x}_1);\langle\tilde{b}_2\rangle;\ldots;(\tilde{x}_{n-1});\langle\tilde{b}_n\rangle;P$

  $\overline{a}\langle\tilde{b}_1\rangle;(\tilde{x}_2);\ldots;\langle\tilde{b}_{n-1}\rangle;(\tilde{x}_n);P$

Branching/Selection

- Branching/Selection

$$P \quad ::= \quad a[(x_1).P_1 \& (x_2).P_2] \mid\ !a[(x_1).P_1 \& (x_2).P_2]$$
$$\mid \quad \overline{a}\,\mathtt{inl}\langle b\rangle.P \mid \overline{a}\,\mathtt{inr}\langle b\rangle.P \cdots$$

- $a[(x_1).P_1 \& (x_2).P_2] | \overline{a}\,\mathtt{inl}\langle b\rangle.Q \longrightarrow P_1\{b/x_1\} \mid Q$

  $a[(x_1).P_1 \& (x_2).P_2] | \overline{a}\,\mathtt{inr}\langle b\rangle.Q \longrightarrow P_2\{b/x_2\} \mid Q$

- Mapping $(\ )^\circ$: Branching/Selection $\pi \rightarrow$ Polyadic π

  $(a[(x_1).P_1 \& (x_2).P_2])^\circ =$
  $$a(c).\overline{c}(c_1 c_2).(c_1(x_1).P_1^\circ \mid c_2(x_2).P_2^\circ)$$

  $(\overline{a}\,\mathtt{inl}\langle b\rangle.Q)^\circ = \overline{a}(c).c(c_1 c_2).\overline{c_1}\langle b\rangle.Q^\circ$

Branching/Selection

- Boolean Agent:

$$\mathtt{Tru}(a) = !a(x).\mathtt{inl}\langle\rangle \qquad \mathtt{Fls}(a) = !a(x).\mathtt{inr}\langle\rangle$$

- If-Then-Else:

$$\mathtt{If}\ a\ \mathtt{then}\ P\ \mathtt{else}\ Q = \overline{a}(c)c[().P\ \&\ ().Q]$$

- $\mathtt{If}\ a\ \mathtt{then}\ P\ \mathtt{else}\ Q \mid \mathtt{Tru}(a) \longrightarrow P$

$$\mathtt{If}\ a\ \mathtt{then}\ P\ \mathtt{else}\ Q \mid \mathtt{Fls}(a) \longrightarrow Q$$

- $(a[().P\&().Q])^\circ = a(c).\overline{c}(c_1c_2)(c_1.P^\circ \mid c_2.Q^\circ)$

$$(\overline{a}\mathtt{inl}\langle\rangle)^\circ = \overline{a}(c)c(c_1c_2).\overline{c_1}$$

$$(\overline{a}\mathtt{inr}\langle\rangle)^\circ = \overline{a}(c)c(c_1c_2).\overline{c_2}$$

Are you fed up with *hacking*
with many name passing?

Time for *Session Types*!

Towards Structured Interactions: Sessions

- offer flexible programming style for structured interaction in communication-centric distributed software.
- statically check safe and consistent compositions of protocols (can be done at run-time or by type inference)

Related Work: Session Types (1)

- Structured Concurrent Languages (Takeuchi, Honda and Kubo) [PARL94]
- Higher-Order Session (Honda, Vasconcelos and Kubo) [ESOP98]
- Subtyping (Gay and Hole) [ESOP00, Acta Informatica 05]
- COLBA Interface (Vallecillo et al) [FOCLASA02]
- Concurrent Haskell (Neubauer and Thiemann) [PADL04]

Related Work: Session Types (2)

- Multi-threaded Functional Languages (Vasconcelos, Ravara and Gay) [CONCUR04]
- Correspondence Assertions (Bonelli, Comagnoni and Gunter) [JFP05]
- Distributed Java (Dezani, Yoshida, Ahern and Drossopoulou) [TCG05]
- Web Service Description Languages (W3C CDL Working Group)
- Microsoft Singularity Operating System (Fähndrich et. al) [EuroSys06]

Related Work: Session Types (3)

- Multi-threaded Concurrent Java (Dezani, Mostrous, Yoshida and Drossopoulou [ECOOP06]
- Formalisation of Web Service Description Languages (Carbone, Honda and Yoshida) [DCM06]
- Analysis of Past Session Typing Systems (Yoshida and Vasconcelos) [SeCReT06]
- Session Types for Ambients (Compagnoni, Dezani and Garralda) [PPDP06]

Session Primitives

- Two Kinds of Usage of Channels

  Shared $(a, b, d, e, \ldots)$ and Session $(c, k, \ldots)$

- Expressions $(e, e', ..)$ e.g. $3 + 1$, etc.

- Processes $(P, Q, ..)$

$\overline{a}(k).P$	$a(k).P$,	$!a(k).P$	initiation
$\overline{k}\langle e_1 \cdots e_n \rangle; P$	$!k(x_1 \cdots x_n); P$		data
$k \lhd l; P$	$k \rhd \{l_1 : P_1 [] \cdots [] l_n : P_n\}$		label
$\overline{k}\langle k' \rangle; P$	$k(k'); P$		delegation

Session Primitives

- Open Session

$$a(k).P_1 \mid \overline{a}(k).P_2 \;\rightarrow\; (\nu k)(P_1 \mid P_2)$$

- Data Exchange (e includes shared names)

$$\overline{k}\langle\tilde{e}\rangle;P_1 \mid\; k(\tilde{x});P_2 \;\rightarrow\; P_1 \mid P_2[\tilde{v}/\tilde{x}] \text{ with } e_i \rightarrow^* v_i$$

- Branching and Selection

$$k \lhd l_i;P \mid k \rhd \{l_1 : P_1 [\!] \cdots [\!] l_n : P_n\} \;\rightarrow\; P \mid\; P_i$$

- Delegation

$$\overline{k}\langle k'\rangle;P_1 \mid\; k(k');P_2 \;\rightarrow\; P_1 \mid\; P_2$$

Bad Interaction (Untypable Terms)

- Base Type Error

  $\overline{k}\langle apple\rangle;P_1 \mid k(x);\overline{k'}\langle 1+x\rangle$

- Arity Mismatch

  $\overline{k}\langle 1\rangle;P_1 \mid k(x,y);\overline{k'}\langle x+y\rangle$

- Break Linearity

  - $k(x);P_1 \mid \overline{k}\langle v\rangle;P_2 \mid \overline{k}\langle w\rangle;P_3$
  - $k(x);\overline{k}\langle w\rangle;\mathbf{0} \mid \overline{k}\langle v\rangle;\mathbf{0}$

- $a(k).P_1 \mid a(k).P_2 \mid \overline{a}(k).P_3 \mid \overline{a}(k).P_4$

Session Types

- Sorts and Types

$$S ::= \mathsf{nat} \mid \mathsf{bool} \mid \langle \alpha, \overline{\alpha} \rangle$$

$$\alpha ::= \downarrow\tilde{S}; \alpha \mid \downarrow\alpha; \beta \mid \&\{l_1 : \alpha_1, \ldots, l_n : \alpha_n\} \mid \mathsf{end} \mid \bot$$
$$\mid \uparrow\tilde{S}; \alpha \mid \uparrow\alpha; \beta \mid \oplus\{l_1 : \alpha_1, \ldots, l_n : \alpha_n\} \mid t \mid \mu t.\alpha$$

- $\overline{\alpha}$ (*Co-type* of α)

$$\overline{\uparrow\tilde{S}; \alpha} = \downarrow\tilde{S}; \overline{\alpha} \qquad \overline{\oplus\{l_i : \alpha_i\}} = \&\{l_i : \overline{\alpha_i}\}$$

$$\overline{\uparrow\alpha; \beta} = \downarrow\alpha; \overline{\beta} \quad \overline{\mathsf{end}} = \mathsf{end} \quad \overline{t} = t \quad \overline{\mu t.\alpha} = \mu t.\overline{\alpha}$$

Session Types

$$\Gamma \vdash P \triangleright \Delta$$

Shared $(a : S, b : S', \ldots)$ Linear $(k : \alpha, k' : \beta, \ldots)$

Key Point a composition of Δ_1 and Δ_2 is defined if all common channels (k in $S = \mathrm{dom}(\Delta_1) \cap \mathrm{dom}(\Delta_2)$) are dual.

$$k(x);\mathbf{0} \quad | \quad \overline{k}\langle v\rangle \quad | \quad \overline{k}\langle w\rangle$$

$$k : \alpha \qquad k : \overline{\alpha} \qquad k : \overline{\alpha}$$

$$k : \bot \qquad k : \overline{\alpha}$$

$$\Delta_1 \circ \Delta_2 \; = \; \{k : \bot \mid k \in S\} \cup (\Delta_1 \cup \Delta_2) \setminus S$$

Typing System

- Base

$$\Gamma \cdot a : S \vdash a \triangleright S \quad \Gamma \vdash 1 \triangleright \mathsf{nat} \quad \frac{\Gamma \vdash e_i \triangleright \mathsf{nat}}{\Gamma \vdash e_1 + e_2 \triangleright \mathsf{nat}}$$

- Nil $\quad \Gamma \vdash \mathbf{0} \triangleright \Delta$ where Δ's codomain is $\perp$ or end.

- Session Initialisation

$$\frac{\Gamma \vdash a \triangleright \langle \alpha, \overline{\alpha} \rangle \quad \Gamma \vdash P \triangleright \Delta \cdot k : \alpha}{\Gamma \vdash a(k).P \triangleright \Delta}$$

$$\frac{\Gamma \vdash a \triangleright \langle \alpha, \overline{\alpha} \rangle \quad \Gamma \vdash P \triangleright \Delta \cdot k : \overline{\alpha}}{\Gamma \vdash \overline{a}(k).P \triangleright \Delta}$$

- Data Passing

$$\frac{\Gamma \vdash \tilde{e} \triangleright \tilde{S} \quad \Gamma \vdash P \triangleright \Delta \cdot k : \alpha}{\Gamma \vdash \overline{k}\langle \tilde{e} \rangle; P \triangleright \Delta \cdot k : \uparrow \tilde{S}; \alpha} \qquad \frac{\Gamma \cdot \tilde{x} : \tilde{S} \vdash P \triangleright \Delta \cdot k : \alpha}{\Gamma \vdash k(\tilde{x}); P \triangleright \Delta \cdot k : \downarrow \tilde{S}; \alpha}$$

- Session over Session

$$\frac{\Gamma \vdash P \triangleright \Delta \cdot k : \beta}{\Gamma \vdash \overline{k}\langle k' \rangle; P \triangleright \Delta \cdot k : \uparrow \alpha; \beta \cdot k' : \alpha}$$

$$\frac{\Gamma \vdash P \triangleright \Delta \cdot k : \beta \cdot k' : \alpha}{\Gamma \vdash k(k'); P \triangleright \Delta \cdot k : \downarrow \alpha; \beta}$$

Branching/Selection

$$\frac{\Gamma \vdash P_1 \triangleright \Delta \cdot k : \alpha_1 \quad \cdots \quad \Gamma \vdash P_n \triangleright \Delta \cdot k : \alpha_n}{\Gamma \vdash\ k \triangleright \{l_1 : P_1 [\!] \cdots [\!] l_n : P_n\} \triangleright \Delta \cdot k : \&\{l_1 : \alpha_1, \ldots, l_n : \alpha_n\}}$$

$$\frac{\Gamma \vdash P \triangleright \Delta \cdot k : \alpha_j}{\Gamma \vdash k \triangleleft l_j ; P \triangleright \Delta \cdot k : \oplus\{l_1 : \alpha_1, \ldots, l_n : \alpha_n\}}$$

Parallel

$$\frac{\Gamma \vdash P \triangleright \Delta \quad \Gamma \vdash Q \triangleright \Delta'}{\Gamma \vdash P \mid Q \triangleright \Delta \circ \Delta'} (\Delta \asymp \Delta')$$

Others

$$\frac{\Gamma \cdot a : S \vdash P \triangleright \Delta}{\Gamma \vdash (\nu a) P \triangleright \Delta} \qquad \frac{\Gamma \vdash P \triangleright \Delta \cdot k : \perp}{\Gamma \vdash\ (\nu k) P \triangleright \Delta} \qquad \frac{\Gamma \vdash P \triangleright \Delta \cdot k : \mathsf{end}}{\Gamma \vdash P \triangleright \Delta \cdot k : \perp}$$

Theorems

1. *(Subject Congruence)*

   $\Gamma \vdash P \triangleright \Delta$ and $P \equiv Q$ imply $\Gamma \vdash Q \triangleright \Delta$.

2. *(Subject Reduction)*

   $\Gamma \vdash P \triangleright \Delta$ and $P \longrightarrow^{*} Q$ imply $\Gamma \vdash Q \triangleright \Delta$.

3. *(Lack of Run-Time Errors)*

   A typable program never reduces into an error.

Typing (1) Branching and Selection

$\Gamma = a : \langle \alpha, \overline{\alpha} \rangle, e : \langle \uparrow \mathtt{string}, \downarrow \mathtt{string} \rangle, d : \langle \uparrow \mathtt{nat}, \downarrow \mathtt{nat} \rangle$

$\alpha = \oplus \{\mathsf{true}, \mathsf{false}\}$

$$\frac{\Gamma \vdash \mathbf{0} \triangleright \emptyset}{\dfrac{\Gamma \vdash a : \langle \alpha, \overline{\alpha} \rangle \quad \Gamma \vdash k \lhd \mathsf{true} \triangleright k : \alpha}{\Gamma \vdash !a(k).k \lhd \mathsf{true} \triangleright \emptyset}}$$

Typing (1) Branching and Selection

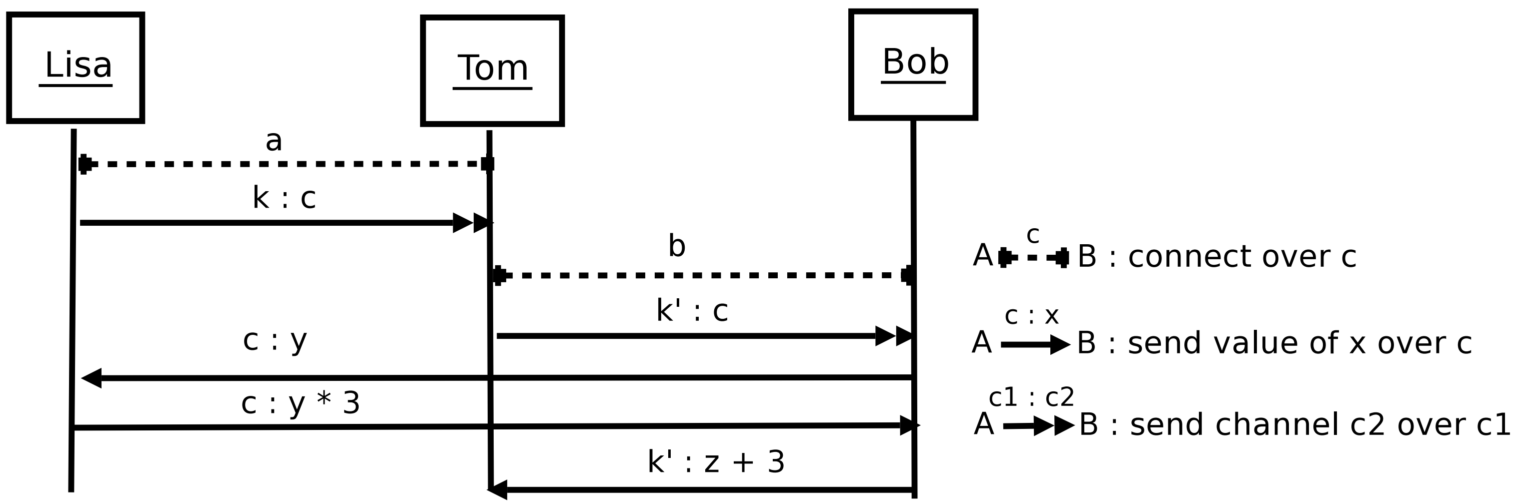
$$\frac{\dfrac{\Gamma \vdash \overline{e}\langle apple \rangle \ \triangleright\ \emptyset \qquad \Gamma \vdash \overline{d}\langle 1 \rangle \ \triangleright\ \emptyset}{\begin{array}{l}\Gamma \vdash a : \langle \alpha, \overline{\alpha} \rangle \\ \Gamma \vdash k \triangleright \{\mathsf{true} : \overline{e}\langle apple \rangle \ [\!]\ \mathsf{false} : \overline{d}\langle 1 \rangle\} \ \triangleright\ k : \overline{\alpha}\end{array}}}{\Gamma \vdash \overline{a}(k).k \triangleright \{\mathsf{true} : \overline{e}\langle apple \rangle \ [\!]\ \mathsf{false} : \overline{d}\langle 1 \rangle\} \ \triangleright\ \emptyset}$$

Typing (2) Delegation

$$\overline{a}(k).\overline{k}\langle c\rangle;c(y);\overline{c}\langle y\times 3\rangle$$

$$a(k).k(c);\overline{b}(k').\overline{k'}\langle c\rangle;k'(y);\overline{e}\langle y+100\rangle$$

$$b(k').k'(c);\overline{c}\langle 2\rangle;c(z);\overline{k'}\langle z+3\rangle$$

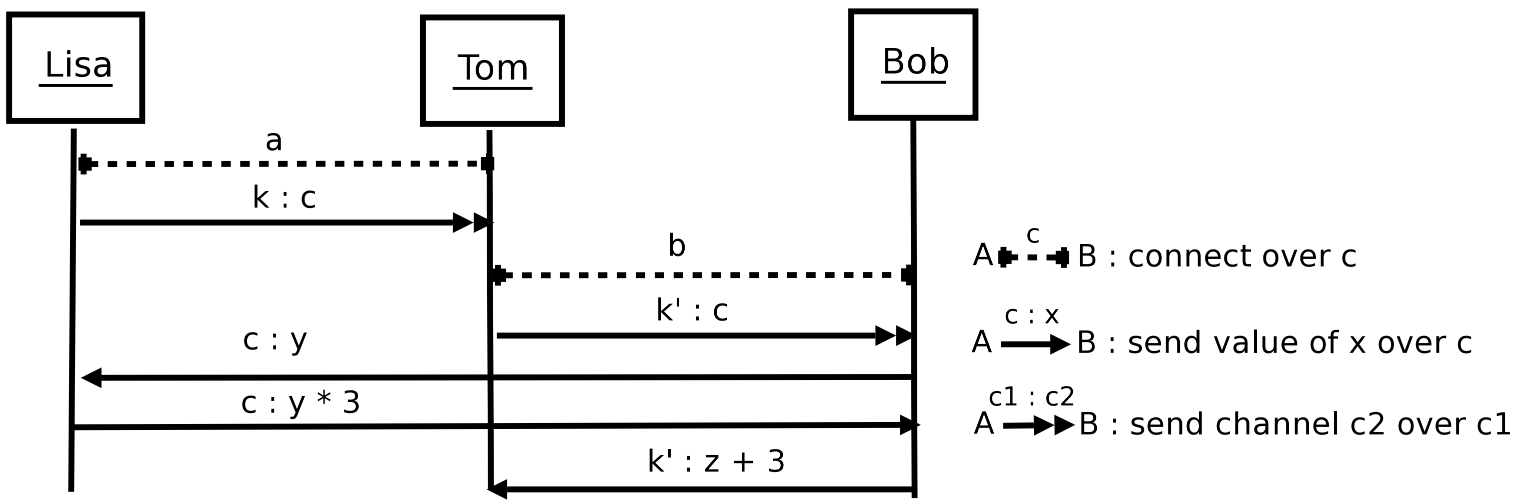


Typing (2) Delegation

$$\overline{a}(k).\overline{k}\langle c\rangle;c(y);\langle y\times 3\rangle$$

$$a(k).k(c);\overline{b}(k').\overline{k'}\langle c\rangle;(y);\overline{e}\langle y+100\rangle$$

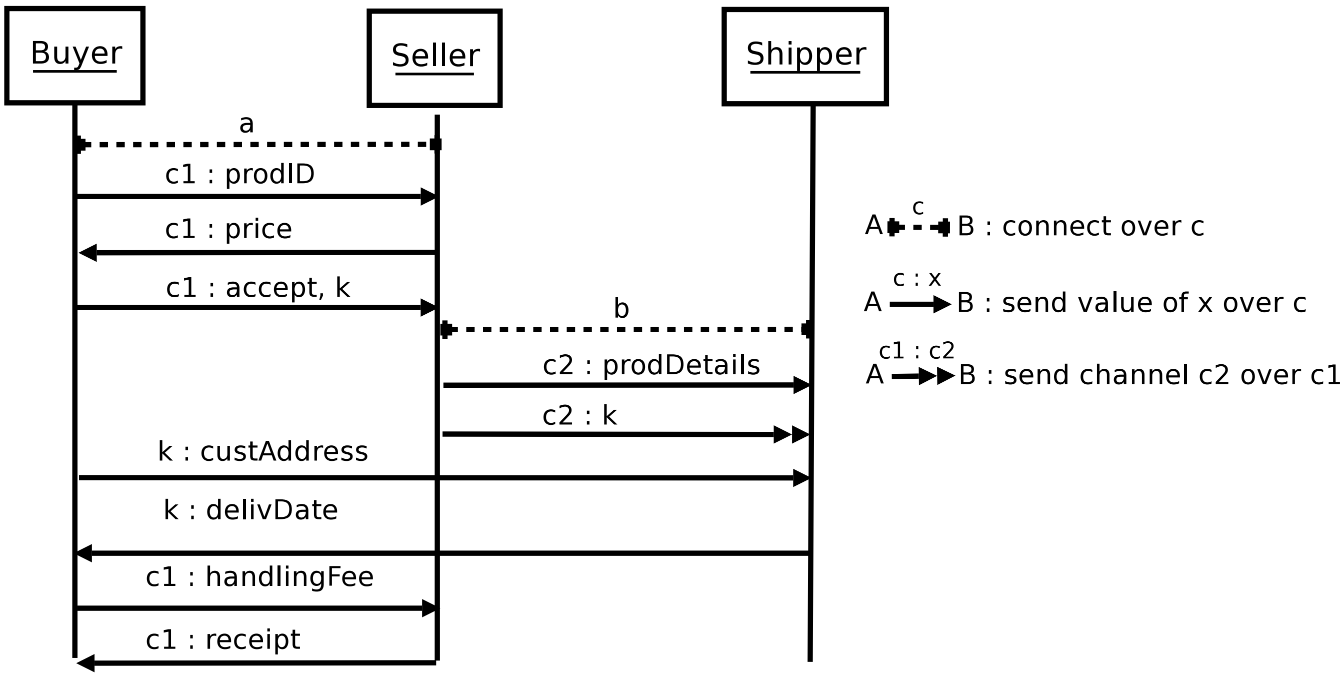
$$b(k').k'(c);\overline{c}\langle 2\rangle;(z);\overline{k'}\langle z+3\rangle$$



Typing (2) Delegation

$$\dfrac{\dfrac{\dfrac{\dfrac{b:\langle\alpha,\overline{\alpha}\rangle, z:\mathsf{nat} \vdash \overline{k'}\langle z+3\rangle \;\triangleright\; k':\uparrow\mathsf{nat}}{b:\langle\alpha,\overline{\alpha}\rangle \vdash c(z);\overline{k'}\langle z+3\rangle \;\triangleright\; c:\downarrow\mathsf{nat},\; k':\uparrow\mathsf{nat}}}{b:\langle\alpha,\overline{\alpha}\rangle \vdash \overline{c}\langle 2\rangle;c(z);\overline{k'}\langle z+3\rangle \;\triangleright\; c:\uparrow\mathsf{nat};\downarrow\mathsf{nat},\; k':\uparrow\mathsf{nat}}}{b:\langle\alpha,\overline{\alpha}\rangle \vdash k'(c);\overline{c}\langle 2\rangle;c(z);\overline{k'}\langle z+3\rangle \;\triangleright\; k':\downarrow(\uparrow\mathsf{nat};\downarrow\mathsf{nat});\uparrow\mathsf{nat}}}{b:\langle\alpha,\overline{\alpha}\rangle \vdash b(k').k'(c);\overline{c}\langle 2\rangle;c(z);\overline{k'}\langle z+3\rangle \;\triangleright\; \emptyset}$$

Protocol Example

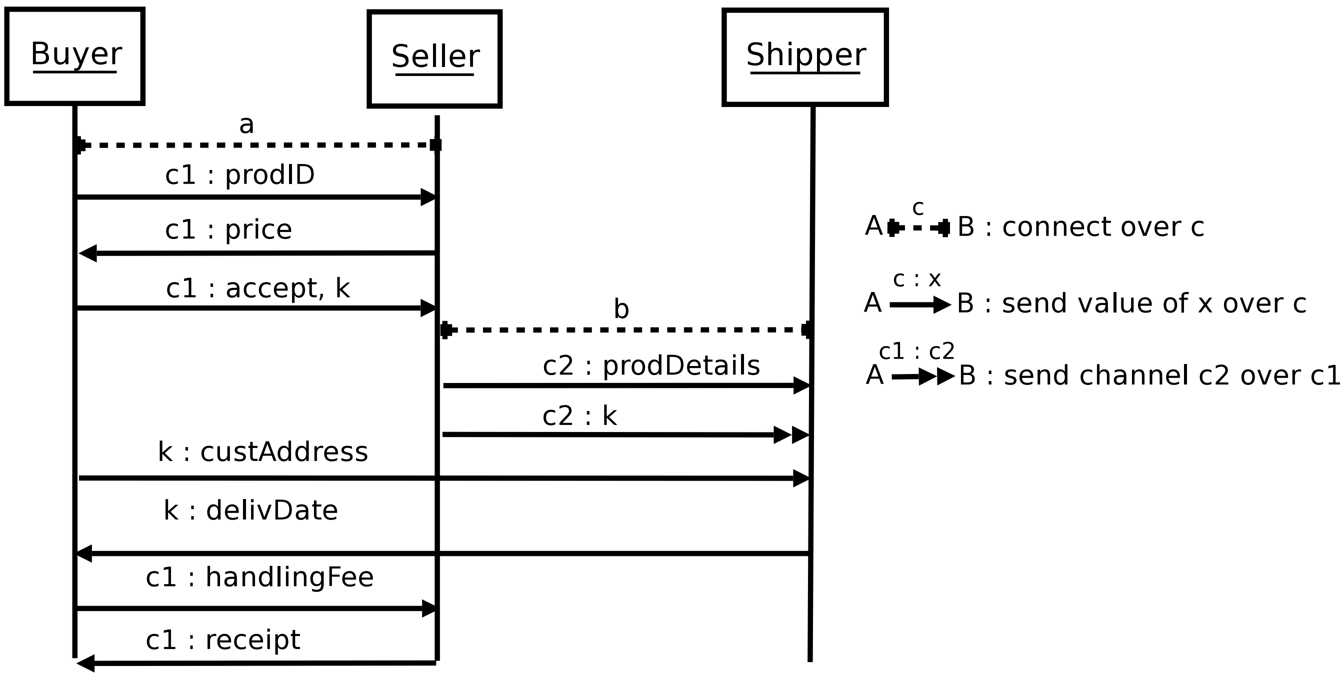


$$\uparrow \oplus \{id : \downarrow double; \oplus \{accept : \uparrow \beta; \uparrow double; \downarrow receipt,\ reject\}\}$$

$$\beta = \uparrow address; \downarrow goods$$

Buyer's viewpoint of the Buyer-Seller interaction

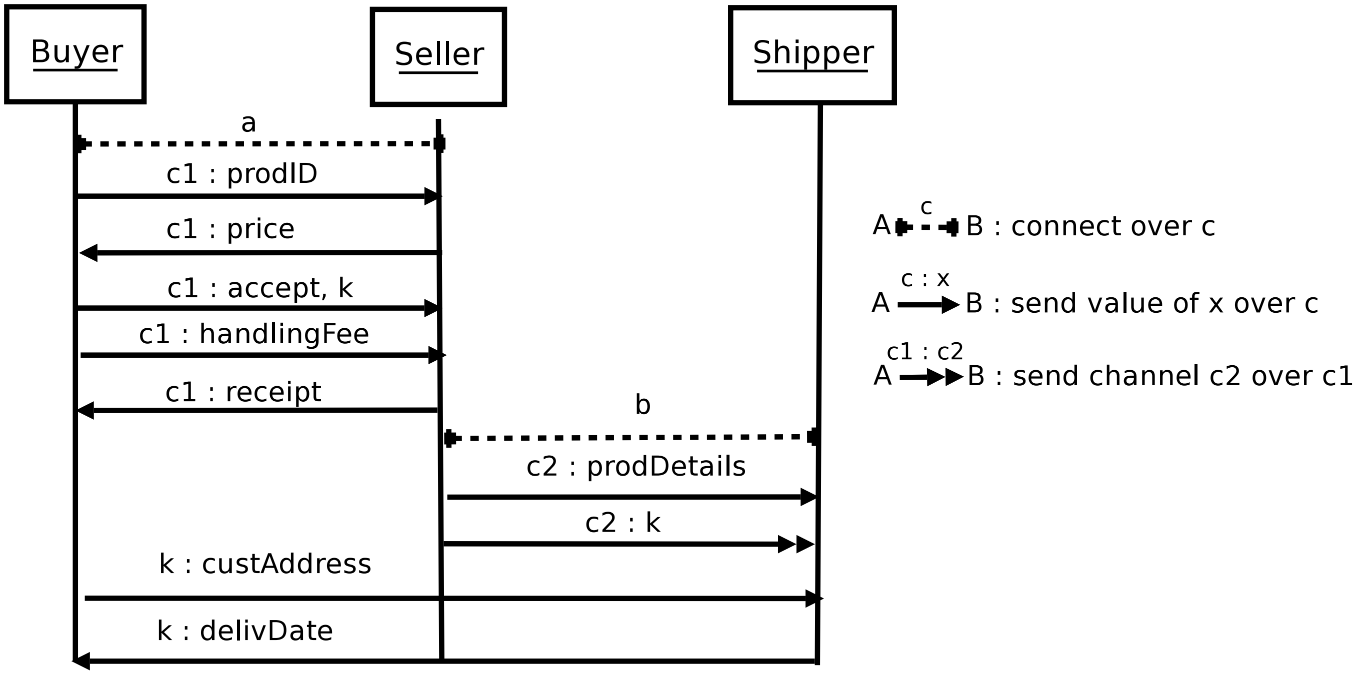
Protocol Example



$$\uparrow \oplus \{\mathsf{id} : \uparrow \beta\}$$

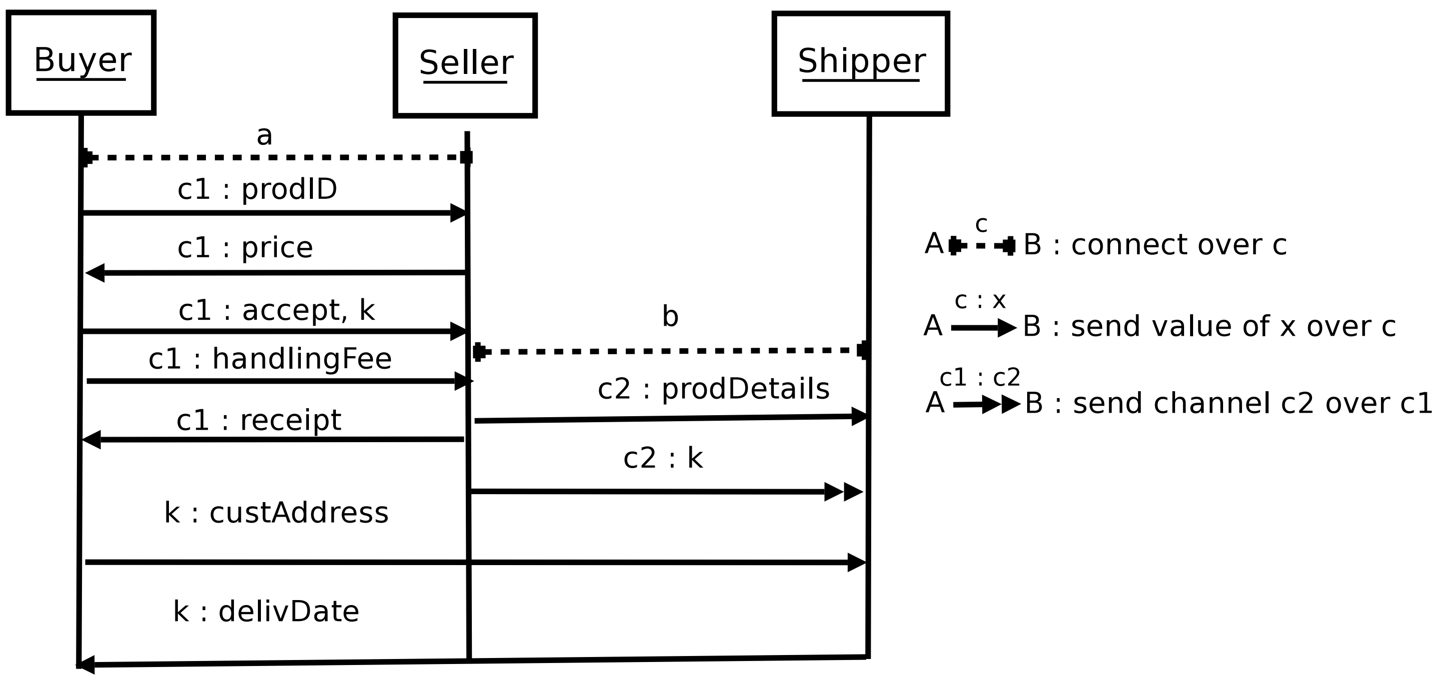
Seller’s viewpoint of the Seller-Shipper interaction

Protocol Example (2): Modest Buyer



Type unchanged

Protocol Example (3): More concurrency



Type unchanged

End-Point Processes (1)

Buyer

$\overline{a}(c_1).c_1 \lhd \mathsf{id}; c_1(y);$

$\quad \texttt{if}\ y < 100\ \texttt{then}$

$\quad c_1 \lhd \mathsf{accept}; \overline{c_1}\langle k \rangle; \overline{k}\langle \texttt{Address} \rangle; k(y); \overline{c_1}\langle 100 \rangle; c_1(z); P$

$\qquad \texttt{else}$

$\quad c_1 \lhd \mathsf{reject};$

End-Point Processes (1)

Buyer

$$\overline{a}(c_1).c_1 \lhd \mathsf{id}(y);$$

$$\texttt{if}\ y < 100\ \texttt{then}$$

$$c_1 \lhd \mathsf{accept}\langle k\rangle;\overline{k}\langle \texttt{Address}\rangle;(y);\overline{c_1}\langle 100\rangle;(z);P$$

$$\texttt{else}$$

$$c_1 \lhd \mathsf{reject};$$

End-Point Processes (2)

Seller

$$a(c_1).c_1 \triangleright \{\mathsf{id} : \overline{c_1}\langle 10\rangle;$$

$$c_1 \triangleright \{\mathsf{accept} : c_1(k);$$

$$\overline{b}(c_2).c_2 \triangleleft \mathsf{id}; \overline{c_2}\langle k\rangle; c_1(y); \overline{c_1}\langle \mathtt{receipt}\rangle$$

$$[\!] \; \mathsf{reject} : Q\}\}$$

End-Point Processes (2)

Seller

$$
\begin{aligned}
& a(c_1).c_1 \triangleright \{\mathsf{id}\langle 10\rangle; \\
& \quad c_1 \triangleright \{\mathsf{accept}(k) \\
& \qquad\qquad\qquad \overline{b}(c_2).c_2 \triangleleft \mathsf{id}\langle k\rangle; c_1(y); \langle \mathtt{receipt}\rangle; \\
& \qquad\quad [\!]\ \mathsf{reject} : Q\}\}
\end{aligned}
$$

End-Point Processes (3)

Modest Buyer

$$\overline{a}(c_1).c_1 \triangleleft \mathsf{id}; c_1(y);$$

$$\texttt{if}\ y < 100\ \texttt{then}$$

$$c_1 \triangleleft \mathsf{accept}; \overline{c_1}\langle k \rangle; \overline{c_1}\langle 100 \rangle; c_1(z); \overline{k}\langle \texttt{Address} \rangle; k(y); P$$

$$\texttt{else}$$

$$c_1 \triangleleft \mathsf{reject};$$

End-Point Processes (3)

Modest Buyer

$\overline{a}(c_1).c_1 \lhd \mathsf{id}(y);$

$\texttt{if } y < 100 \texttt{ then}$

$c_1 \lhd \mathsf{accept}\langle k\rangle; \langle 100\rangle; (z); \overline{k}\langle \texttt{Address}\rangle; (y); P$

$\texttt{else}$

$c_1 \lhd \mathsf{reject};$

Observations

- Diagrams are not precise, but the end-point behaviour is precise
- But each end-point behaviour is still very fine-grained, contains too much information, and is inconvenient for programmers to *directly* write a *global scenario*.

Conclusion

- The π-Calculus
- Idioms for Interactions
- Session Types

Part 2 Web Services and the π-Calculus

- 1 Web Services Choreography Description Language
- 2 Global Language and the End-Point Calculus
- 3 Correctness

References

- References **www.doc.ic.ac.uk/∼yoshida/tic/**
- The π-Calculus
  - The π-Calculus: a Theory of Mobile Processes (CUP)
    Davide Sangiorgi and David Walker
  - The π-Calculus (CUP)   Robin Milner
- Session Types
  - Language Primitives and Type Discipline for
    Structured Communication-Based Programming
    Honda, Vasconcelos and Kubo [ESOP98]
  - Revisit, Vasconcelos and Yoshida [SecReT06]