9 Months Report

A Higher Order Client-Server Calculus with Sessions

Dimitris Mostrous

Supervisor Nobuko Yoshida

December 11, 2006

Overview

Topic

- how to structure and describe protocols in concurrent/distributed programming
- methods must be tractable

Method

- foundational calculus (higher-order π) with sessions
  - hopi can encode a large class of programs/idioms (objects, mobile code, closures etc)
  - sessions can describe a large class of protocols

Introduction

A session describes a communication protocol between <u>two</u> parties, that takes place over a <u>single</u> connection

Kohei Honda. *Types for Dyadic Interaction*.
CONCUR'93, LNCS 715, pages 509–523, Springer-Verlag, 1993

Introduction

A session describes a communication protocol between two parties, that takes place over a single connection

Kohei Honda. *Types for Dyadic Interaction*.
CONCUR'93, LNCS 715, pages 509–523, Springer-Verlag, 1993

We previously integrated sessions in a small object calculus

Mariangiola Dezani-Ciancaglini, Dimitris Mostrous, Nobuko Yoshida, and Sophia Drossopoulou. *Session Types for Object-Oriented Languages*. ECOOP 2006.

Example

Session type for channel a:

$$a \ : \ \text{begin}.![\mathit{int}].![\mathit{int}].?[\mathit{bool}].\text{end}$$

agrees with $\mathrm{HO}\pi^s$ code:

$$P = \overline{a}(x).\overline{x}\langle 5\rangle.\overline{x}\langle 5\rangle.x(y{:}\mathit{bool})$$

intuition: a is a network address; x is the io stream

Example

Session type for channel a:

$$a \; : \; \text{begin}.![int].![int].?[bool].\text{end}$$
$$a \; : \; \text{begin}.?[int].?[int].![bool].\text{end}$$

agrees with $\text{HO}\pi^s$ code:

$$P = \overline{a}(x).\overline{x}\langle 5 \rangle.\overline{x}\langle 5 \rangle.x(y:bool)$$

$$Q = !a(x).x(y:int).x(z:int).\overline{x}\langle \mathsf{T} \rangle$$

$$P \mid Q \text{ is typable}$$

The Types

Types:

$$\sigma ::= \mathsf{begin}.\rho \qquad \varphi ::= \tau \to \psi \qquad \zeta ::= \mathsf{unit} \mid \sigma \mid \varphi$$

$$\tau ::= \zeta \mid (\varphi)^1 \mid \rho \qquad \psi ::= \tau \mid \diamond$$

$$\rho ::= ![\tau].\rho \mid ?[\tau].\rho \mid \&[l_1:\rho_1;\ldots;l_n:\rho_n] \mid \oplus[l_1:\rho_1;\ldots;l_n:\rho_n] \mid \mathsf{end}$$

Duality: $\overline{![\tau].\rho} = ?[\tau].\overline{\rho} \qquad \overline{?[\tau].\rho} = ![\tau].\overline{\rho} \qquad \overline{\mathsf{end}} = \mathsf{end}$

$$\overline{\&[l_1:\rho_1;\ldots;l_n:\rho_n]} = \oplus[l_1:\overline{\rho_1};\ldots;l_n:\overline{\rho_n}]$$

$$\overline{\oplus[l_1:\rho_1;\ldots;l_n:\rho_n]} = \&[l_1:\overline{\rho_1};\ldots;l_n:\overline{\rho_n}]$$

The Types

Types:

$$\sigma ::= \mathsf{begin}.\rho \qquad \varphi ::= \tau \rightarrow \psi \qquad \zeta ::= \mathsf{unit} \mid \sigma \mid \varphi$$

$$\tau ::= \zeta \mid (\varphi)^1 \mid \rho \qquad \psi ::= \tau \mid \diamond$$

$$\rho ::= ![\tau].\rho \mid ?[\tau].\rho \mid \&[l_1 : \rho_1; \ldots ; l_n : \rho_n] \mid \oplus[l_1 : \rho_1; \ldots ; l_n : \rho_n] \mid \mathsf{end}$$

Duality:

$$\overline{![\tau].\rho} = ?[\tau].\overline{\rho} \qquad \overline{?[\tau].\rho} = ![\tau].\overline{\rho} \qquad \overline{\mathsf{end}} = \mathsf{end}$$

$$\overline{\&[l_1 : \rho_1; \ldots ; l_n : \rho_n]} = \oplus[l_1 : \overline{\rho_1}; \ldots ; l_n : \overline{\rho_n}]$$

$$\overline{\oplus[l_1 : \rho_1; \ldots ; l_n : \rho_n]} = \&[l_1 : \overline{\rho_1}; \ldots ; l_n : \overline{\rho_n}]$$

Operational Semantics I

Structural rules:

$$(\textsc{str}) : \frac{P \equiv P' \longrightarrow Q' \equiv Q}{P \longrightarrow Q}$$

$$(\textsc{par}) : \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$$

$$(\textsc{res}) : \frac{P \longrightarrow P'}{(\nu \tilde{a} : \tilde{\sigma})(\nu \tilde{\kappa})P \longrightarrow (\nu \tilde{a} : \tilde{\sigma})(\nu \tilde{\kappa})P'}$$

Operational Semantics I

Structural rules:

$$(\textsc{str}) : \frac{P \equiv P' \longrightarrow Q' \equiv Q}{P \longrightarrow Q}$$

$$(\textsc{par}) : \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$$

$$(\textsc{res}) : \frac{P \longrightarrow P'}{(\nu \tilde{a} : \tilde{\sigma})(\nu \tilde{\kappa})P \longrightarrow (\nu \tilde{a} : \tilde{\sigma})(\nu \tilde{\kappa})P'}$$

Operational Semantics II

Functional rules:

$$(\text{APP-L}) : \frac{P \longrightarrow P'}{P \cdot Q \longrightarrow P' \cdot Q} \qquad (\text{APP-R}) : \frac{Q \longrightarrow Q'}{V \cdot Q \longrightarrow V \cdot Q'}$$

$$(\beta) : \frac{}{(\lambda(x : \tau).P) \cdot V \longrightarrow P\{V/x\}}$$

Operational Semantics II

Functional rules:

$$(\textsf{APP-L}) : \frac{P \longrightarrow P'}{P \cdot Q \longrightarrow P' \cdot Q} \qquad (\textsf{APP-R}) : \frac{Q \longrightarrow Q'}{V \cdot Q \longrightarrow V \cdot Q'}$$

$$(\beta) : \frac{}{(\lambda(x:\tau).P) \cdot V \longrightarrow P\{{}^{V}\!/_{x}\}}$$

Operational Semantics III

Session rules:

$$(\textsf{CONN}) : \frac{}{!a(x).P \mid \overline{a}(z).Q \longrightarrow !a(x).P \mid (\nu\kappa)\,(P\{^{\kappa_0}/_x\} \mid Q\{^{\kappa_1}/_z\})}$$

$$(\textsf{COMM}) : \frac{}{\kappa_i(x).P \mid \overline{\kappa}_j\langle V\rangle.Q \longrightarrow P\{^V/_x\} \mid Q} \quad i+j=1$$

$$(\textsf{LABEL}) : \frac{}{\kappa_i \lhd l_m.P \mid \kappa_j \rhd \{l_1 : P_1; \ldots; l_n : P_n\} \longrightarrow P \mid P_m}$$

$$i+j=1,\ 1 \leq m \leq n$$

Operational Semantics III

Session rules:

$$(\textsc{conn}): \frac{}{!a(x).P \mid \overline{a}(z).Q \longrightarrow !a(x).P \mid (\nu\kappa)\,(P\{\kappa_0/x\} \mid Q\{\kappa_1/z\})}$$

$$(\textsc{comm}): \frac{}{\kappa_i(x).P \mid \overline{\kappa}_j\langle V\rangle.Q \longrightarrow P\{V/x\} \mid Q}\ \ i+j=1$$

$$(\textsc{label}): \frac{}{\kappa_i \lhd l_m.P \mid \kappa_j \rhd \{l_1 : P_1; \ldots ; l_n : P_n\} \longrightarrow P \mid P_m}$$

$$i+j=1,\ 1 \leq m \leq n$$

Typing System

Key Points

- Usage of session channels must agree with type;
- session channels must not become forgotten
  $(\,\lambda(x:\rho).\mathbf{0}\,)\cdot\kappa_0$
- session channels must not be copied
  $(\,\lambda(x:\rho).(\overline{x}\langle V\rangle \mid \overline{x}\langle V'\rangle)\,)\cdot\kappa_0$
- abstractions that contain running sessions must be used exactly once
  $(\,\lambda(x:\varphi).x\cdot()\,)\cdot\lambda(z:\text{unit}).(\,\kappa_1\langle 5\rangle.\mathbf{0}\,) \qquad \varphi = \text{unit}\longrightarrow\diamond$
  $(\,\lambda(x:\varphi).\mathbf{0}\,)\cdot\lambda(z:\text{unit}).(\,\kappa_1\langle 5\rangle.\mathbf{0}\,)$
  $(\,\lambda(x:\varphi).\mathbf{0}\,)\cdot\lambda(z:\text{unit}).(\overline{a}(x).\overline{x}\langle 5\rangle.\mathbf{0}\,)$
  $(\,\lambda(z:\varphi).((\lambda(x:\varphi).\mathbf{0})\cdot z)\,)\cdot\lambda(y:\text{unit}).(\,\kappa_1\langle 5\rangle\,)$

Typing System

Key Points

- Usage of session channels must agree with type;
- session channels must not become forgotten
  $(\,\lambda(x:\rho).\mathbf{0}\,)\cdot\kappa_0$
- session channels must not be copied
  $(\,\lambda(x:\rho).(\overline{x}\langle V\rangle \mid \overline{x}\langle V'\rangle)\,)\cdot\kappa_0$
- abstractions that contain running sessions must be used exactly once
  $(\,\lambda(x:\varphi).x\cdot()\,)\;\cdot\;\lambda(z:\text{unit}).(\,\kappa_1\langle 5\rangle.\mathbf{0}\,) \qquad \varphi = \text{unit} \longrightarrow \diamond$
  $(\,\lambda(x:\varphi).\mathbf{0}\,)\;\cdot\;\lambda(z:\text{unit}).(\,\kappa_1\langle 5\rangle.\mathbf{0}\,)$
  $(\,\lambda(x:\varphi).\mathbf{0}\,)\;\cdot\;\lambda(z:\text{unit}).(\overline{a}(x).\overline{x}\langle 5\rangle.\mathbf{0}\,)$
  $(\,\lambda(z:\varphi).((\lambda(x:\varphi).\mathbf{0})\cdot z)\,)\;\cdot\;\lambda(y:\text{unit}).(\,\kappa_1\langle 5\rangle\,)$

Typing System

Typing Judgement

$$\Gamma;\Sigma;\Phi \vdash P : \psi$$

$$\begin{aligned} \Gamma &::= \emptyset \mid \Gamma, a : \sigma \mid \Gamma, x : \zeta \mid \Gamma, x : (\varphi)^1 \\ \Sigma &::= \emptyset \mid \Sigma, k : \rho \\ \Phi &::= \emptyset \mid \Phi, x \end{aligned}$$

– Γ, Γ' and Σ, Σ' are disjoint-domain unions;

– Φ, Φ' is disjoint union;

– $dom(\Sigma) \cap \Phi = \emptyset$ for all well-formed judgements.

Typing System

Typing Judgement

$$\Gamma; \Sigma; \Phi \vdash P : \psi$$

$$\begin{array}{lcl} \Gamma & ::= & \emptyset \mid \Gamma, a : \sigma \mid \Gamma, x : \zeta \mid \Gamma, x : (\varphi)^1 \\ \Sigma & ::= & \emptyset \mid \Sigma, k : \rho \\ \Phi & ::= & \emptyset \mid \Phi, x \end{array}$$

– Γ, Γ' and Σ, Σ' are disjoint-domain unions;

– Φ, Φ' is disjoint union;

– $dom(\Sigma) \cap \Phi = \emptyset$ for all well-formed judgements.

Typing System

(SHARED)

$$\frac{}{\Gamma, u : \zeta; \emptyset; \emptyset \vdash u : \zeta}$$

(UNIT)

$$\frac{}{\Gamma; \emptyset; \emptyset \vdash () : \mathsf{unit}}$$

(LINFUNC)

$$\frac{}{\Gamma, x : (\varphi)^1; \emptyset; \{x\} \vdash x : (\varphi)^1}$$

(LINCHAN)

$$\frac{}{\Gamma; \{k : \rho\}; \emptyset \vdash k : \rho}$$

Typing System

(SUB$_\varphi$)

$$\frac{\Gamma;\Sigma;\Phi \vdash P : \varphi}{\Gamma;\Sigma;\Phi \vdash P : (\varphi)^1}$$

– $\varphi \preceq (\varphi)^1$

– we can derive $\Gamma, x : \varphi; \emptyset; \emptyset \vdash x : (\varphi)^1$

– but $\Gamma, x : (\varphi)^1; \emptyset; \{x\} \vdash x : (\varphi)^1$

Typing System

(ABS)

$$\frac{\Gamma, x:\tau; \Sigma; \Phi \vdash P : \psi \qquad \text{if } \tau = (\varphi)^1, x \in \Phi}{\Gamma; \Sigma; \Phi \setminus x \vdash \lambda(x:\tau).P : \tau \rightarrow \psi}$$

(RCV)

$$\frac{\Gamma, x:\tau; \Sigma, k:\rho; \Phi \vdash P : \diamond \qquad \text{if } \tau = (\varphi)^1, x \in \Phi}{\Gamma; \Sigma, k:?[\tau].\rho; \Phi \setminus x \vdash k(x).P : \diamond}$$

Typing System

(ABS$_\rho$)

$$\frac{\Gamma;\Sigma,x:\rho;\Phi\vdash P:\psi}{\Gamma;\Sigma;\Phi\vdash\lambda(x:\rho).P:\rho\rightarrow\psi}$$

(RCV$_\rho$)

$$\frac{\Gamma;\Sigma,k:\rho',x:\rho;\Phi\vdash P:\diamond}{\Gamma;\Sigma,k:?[\rho].\rho';\Phi\vdash k(x).P:\diamond}$$

Typing System

(App)

$$\frac{\Gamma;\Sigma_1;\Phi_1 \vdash P : \tau \rightarrow \psi \qquad \Gamma;\Sigma_2;\Phi_2 \vdash Q : \tau \qquad \text{if } \tau = \varphi,\ \Sigma_2 = \Phi_2 = \emptyset}{\Gamma;\Sigma_1,\Sigma_2;\Phi_1,\Phi_2 \vdash P \cdot Q : \psi}$$

(Snd)

$$\frac{\Gamma;\Sigma_2;\Phi_2 \vdash P : \diamond \qquad \Gamma;\Sigma_1;\Phi_1 \vdash V : \tau \qquad k : \rho \in \Sigma_{i\in\{1,2\}} \qquad \text{if } \tau = \varphi,\ \Sigma_1 = \Phi_1 = \emptyset}{\Gamma;(\Sigma_1,\Sigma_2)\setminus\{k:\rho\},k:![\tau].\rho;\Phi_1,\Phi_2 \vdash \overline{k}\langle V\rangle.P : \diamond}$$

Typing System

Balanced session environments

$$(\textsc{New}_\kappa)\ \frac{\Gamma;\Sigma,\kappa_0:\rho,\kappa_1:\overline{\rho};\Phi\vdash P:\diamond}{\Gamma;\Sigma;\Phi\vdash(\nu\kappa)P:\diamond}$$

balanced(Σ) if $\Sigma\equiv\{\kappa_0^i:\rho^i\mid i\in I\},\left\{\kappa_1^i:\overline{\rho^i}\mid i\in I\right\}$

Properties

Subject reduction (soundness)

If

– $\Gamma; \Sigma; \Phi \vdash P : \psi$

– $P \longrightarrow P'$

– balanced(Σ)

then

– $\Gamma; \Sigma'; \Phi' \vdash P' : \psi$

– balanced(Σ')

– $\Phi' \subseteq \Phi$

Related work

- Linear λ-calculus. (See Ch. 1 of "Advanced Topics in Types and Programming Languages" by David Walker, ed. B.C. Pierce.)
- Session typing for FO languages. (Many systems, including MOOSE.)
- Higher order π. (PhD Thesis by D. Sangiorgi.)

Planning

$HO\pi^s$

- Finish proofs & Submit paper to TLCA (International Conference on Typed Lambda Calculi and Applications) [January]

Objects

- Finalise journal version of ECOOP’06 paper [January]
- Complete new object calculus, combining objects, $HO\pi$, and sessions [March]

Questions & Further Information

Questions ?

for more information and downloads:

www.doc.ic.ac.uk/∼dm04

dm04@doc

Questions & Further Information

Questions ?

for more information and downloads:

www.doc.ic.ac.uk/∼dm04

dm04@doc

Questions & Further Information

Questions ?

for more information and downloads:

www.doc.ic.ac.uk/∼dm04

dm04@doc