Contracts for Mobile Processes

Giuseppe Castagna   Luca Padovani

Laboratoire PPS, CNRS, Université Paris Diderot

Istituto di Scienze e Tecnologie dell'Informazione, Università di Urbino

Parsec meeting, January 26th, 2009

Outline

1. Motivation
   - Protocols and processes
   - Contracts and mobile systems
2. Contracts
   - Syntax
   - Semantics
3. Results
4. Concluding remarks

Protocols and processes

Session types

- prescriptions on the use of channels

$$u : \sigma, v : \tau, \cdots \vdash P$$

Contracts

- overall process behavior

$$u : \mathtt{Ch}, v : \mathtt{Ch}, \cdots \vdash P : T$$

Summary

- both are behavioral types
- σ = projection of T on u

What session types and contracts are for

Characterizing well-formed systems

- the system eventually terminates
- the system never deadlocks

Characterizing well-typed processes

- sent messages have the correct/expected type
- messages sent/delivered in the right order

Reasoning about processes by means of their type

- refactoring processes
- searching for services

A problem of abstraction

Session types	Contracts
?Int.?Int.(!Real $\oplus$!Error)	$a.a.(\overline{b} \oplus \overline{c})$
?(!Bool.!Bool)	a **?**

A natural candidate

Contracts without channel passing $\Rightarrow$ **CCS**

Contracts with channel passing $\Rightarrow$ π-calculus

A problem of abstraction

Session types	Contracts
`?Int.?Int.(!Real` $\oplus$ `!Error)`	$a.a.(\overline{b} \oplus \overline{c})$
`?(!Bool.!Bool)`	a **?**

A natural candidate

Contracts without channel passing $\Rightarrow$ **CCS**

Contracts with channel passing $\Rightarrow$ π-calculus

An example

process $\mathtt{store}?(\!|x|\!).x?(y : \mathtt{Item}).$
$\mathtt{if}\ y$ is in stock
$\mathtt{then}\ \mathtt{bank}!(\!|x|\!)$
$\mathtt{else}\ x!\langle \mathsf{available}(y) \rangle$

$\Downarrow$

contract $\mathtt{store}?(x).x?\mathtt{Item}.(\mathtt{bank}!x.\mathbf{1} \oplus x!\mathrm{Date}.\mathbf{1})$

An example

process $\texttt{store?}(\!|x|\!).x?(y : \texttt{Item}).$

$\texttt{if}$ y is in stock

$\texttt{then}$ $\texttt{bank!}(\!|x|\!)$

$\texttt{else}$ $x!\langle\text{available}(y)\rangle$

$\Downarrow$

contract $\texttt{store?}(x).x?\texttt{Item}.(\texttt{bank!}x.\mathbf{1} \oplus x!\texttt{Date}.\mathbf{1})$

An example

process $\texttt{store?}(\!|x|\!).x?(y : \texttt{Item}).$
$\quad$ $\texttt{if}$ y is in stock
$\quad\quad$ $\texttt{then bank!}(\!|x|\!)$
$\quad\quad$ $\texttt{else}$ $x!\langle \text{available}(y)\rangle$

$$\Downarrow$$

contract $\texttt{store?}(x).x?\texttt{Item}.(\texttt{bank!}x.\mathbf{1} \oplus x!\texttt{Date}.\mathbf{1})$

An example

process $\mathtt{store}?(\!|x|\!).x?(y : \mathtt{Item}).$

$\quad \mathtt{if}\ y$ is in stock

$\quad\quad \mathtt{then}\ \mathtt{bank}!(\!|x|\!)$

$\quad\quad \mathtt{else}\ x!\langle \mathsf{available}(y)\rangle$

$$\Downarrow$$

contract $\mathtt{store}?(x).x?\mathtt{Item}.(\mathtt{bank}!x.\mathbf{1} \oplus x!\mathtt{Date}.\mathbf{1})$

Some typing rules

V-SEND

$$\frac{\Gamma \vdash e : t \qquad \Gamma \vdash P : T}{\Gamma \vdash \alpha!e.P : \alpha!t.T}$$

V-RECV

$$\frac{\Gamma, x : t \vdash P : T}{\Gamma \vdash \alpha?(x : t).P : \alpha?t.T}$$

C-SEND

$$\frac{\Gamma \vdash P : T}{\Gamma \vdash \alpha!(\!|\beta|\!).P : \alpha!\beta.T}$$

C-RECV

$$\frac{\Gamma, x : \mathtt{Ch} \vdash P : T}{\Gamma \vdash \alpha?(\!|x|\!).P : \alpha?(x).T}$$

Some typing rules

V-SEND
$$\frac{\Gamma \vdash e : t \qquad \Gamma \vdash P : T}{\Gamma \vdash \alpha!e.P : \alpha!t.T}$$

V-RECV
$$\frac{\Gamma, x : t \vdash P : T}{\Gamma \vdash \alpha?(x : t).P : \alpha?t.T}$$

C-SEND
$$\frac{\Gamma \vdash P : T}{\Gamma \vdash \alpha!(\!|\beta|\!).P : \alpha!\beta.T}$$

C-RECV
$$\frac{\Gamma, x : \mathtt{Ch} \vdash P : T}{\Gamma \vdash \alpha?(\!|x|\!).P : \alpha?(x).T}$$

undecidable $\rightarrow$ decidable

Outline

1. Motivation
    - Protocols and processes
    - Contracts and mobile systems
2. Contracts
    - Syntax
    - Semantics
3. Results
4. Concluding remarks

Syntax

failure, success

$$T \;::=\; \mathbf{0} \mid \mathbf{1} \qquad \pi.T \mid T + T \mid T \oplus T \qquad T|T \;\mid\; (\boldsymbol{\nu} a)T$$

$$\pi \;::=\; \alpha?f \mid \alpha!f \mid \alpha!(a)$$

$$f \;::=\; x \mid (x) \mid a \mid \texttt{Int} \mid \texttt{Bool} \mid \cdots$$

Infinite behaviors = infinite terms

- regularity $\qquad X = c?\texttt{Int}.X$
- boundedness $\qquad X = a?(x).(c!x.\mathbf{1} \mid X)$

Syntax

dynamic operators

$$T \;::=\; \mathbf{0} \mid \mathbf{1} \qquad \pi.T \mid T + T \mid T \oplus T \qquad T|T \;\mid\; (\nu a)T$$

$$\pi \;::=\; \alpha?f \mid \alpha!f \mid \alpha!(a)$$

$$f \;::=\; x \mid (x) \mid a \mid \mathtt{Int} \mid \mathtt{Bool} \mid \cdots$$

Infinite behaviors = infinite terms

- regularity $X = c?\mathtt{Int}.X$
- boundedness $X = a?(x).(c!x.\mathbf{1} \mid X)$

Syntax

systems

$$T \; ::= \; \mathbf{0} \mid \mathbf{1} \quad \pi.T \mid T + T \mid T \oplus T \quad T|T \mid (\boldsymbol{\nu} a)T$$

$$\pi \; ::= \; \alpha?f \mid \alpha!f \mid \alpha!(a)$$

$$f \; ::= \; x \mid (x) \mid a \mid \texttt{Int} \mid \texttt{Bool} \mid \cdots$$

Infinite behaviors = infinite terms

- regularity $X = c?\texttt{Int}.X$
- boundedness $X = a?(x).(c!x.\mathbf{1} \mid X)$

Syntax

$$T \;::=\; \mathbf{0} \mid \mathbf{1} \qquad \pi.T \mid T + T \mid T \oplus T \qquad T|T \;\mid\; (\nu a)T$$

$$\pi \;::=\; \alpha?f \mid \alpha!f \mid \alpha!(a)$$ ← prefixes

$$f \;::=\; x \mid (x) \mid a \mid \texttt{Int} \mid \texttt{Bool} \mid \cdots$$

Infinite behaviors = infinite terms

- regularity $\qquad X = c?\texttt{Int}.X$
- boundedness $\qquad X = a?(x).(c!x.\mathbf{1} \mid X)$

Syntax

$$T ::= \mathbf{0} \mid \mathbf{1} \qquad \pi.T \mid T + T \mid T \oplus T \qquad T|T \mid (\nu a)T$$

$$\pi ::= \alpha?f \mid \alpha!f \mid \alpha!(a)$$

$$f ::= x \mid (x) \mid a \mid \mathtt{Int} \mid \mathtt{Bool} \mid \cdots$$

patterns = sets of values and names + binders

Infi

- regularity $X = c?\mathtt{Int}.X$
- boundedness $X = a?(x).(c!x.\mathbf{1} \mid X)$

Syntax

$$T ::= \mathbf{0} \mid \mathbf{1} \qquad \pi.T \mid T + T \mid T \oplus T \qquad T|T \;\mid\; (\boldsymbol{\nu} a)T$$

$$\pi ::= \alpha?f \mid \alpha!f \mid \alpha!(a)$$

$$f ::= x \mid (x) \mid a \mid \texttt{Int} \mid \texttt{Bool} \mid \cdots$$

Infinite behaviors = infinite terms

- regularity $X = c?\texttt{Int}.X$
- boundedness $X = a?(x).(c!x.\mathbf{1} \mid X)$

Labeled operational semantics

$$\mathbf{1} \stackrel{\checkmark}{\longrightarrow} \mathbf{1}$$

$$\frac{m \in f \rightsquigarrow \sigma}{c?f.T \stackrel{c?m}{\longrightarrow} T\sigma} \qquad \frac{m \in f}{c!f.T \longrightarrow c!m.T} \qquad c!m.T \stackrel{c!m}{\longrightarrow} T$$

Example

$20 \in \mathtt{Int}$ $\qquad$ $20 \in \mathtt{Real} \rightsquigarrow \emptyset$

$$c!\mathtt{Int}.\mathbf{1} \mid c?\mathtt{Real}.\mathbf{1} \longrightarrow c!20.\mathbf{1} \mid c?\mathtt{Real}.\mathbf{1} \longrightarrow \mathbf{1} \mid \mathbf{1} \stackrel{\checkmark}{\longrightarrow}$$

Contracts as behavioral types

Systems

$$S \stackrel{\text{def}}{=} T_1 \mid T_2 \mid \cdots \mid T_n$$

1. when is a system well-formed?
2. when is a process well-typed?
3. when are two types equal?

Participant satisfaction

Definition

$T \lhd S$ if $T \mid S \Longrightarrow T' \mid S'$ and $T' \longrightarrow\!\!\!\!\!/\;$ implies

- $T' \xrightarrow{\mu_1}$ and $S' \xRightarrow{\mu_2}$
- $\mu_1 \# \mu_2$ $\qquad (c!m \# c?m,\ \checkmark \# \checkmark)$

for some μ_1 and μ_2

Examples

- $c!\texttt{Int}.\mathbf{1} \lhd c?\texttt{Real}.\mathbf{1}$
- $c!\texttt{Real}.\mathbf{1} \not\lhd c?\texttt{Int}.\mathbf{1}$
  $c!\texttt{Real}.\mathbf{1} \mid c?\texttt{Int}.\mathbf{1} \longrightarrow c!\sqrt{2}.\mathbf{1} \mid c?\texttt{Int}.\mathbf{1}$ stuck

Well-formed systems

$$S \stackrel{\text{def}}{=} T_1 \mid T_2 \mid \cdots \mid T_n$$

Definition

S is *well formed* if $\quad T_k \lhd \prod_{i \in \{1,\dots,n\} \setminus \{k\}} T_i \quad$ for every $1 \leq k \leq n$

Examples

- $c!\mathtt{Int}.\mathbf{1} \mid c?\mathtt{Real}.\mathbf{1}$ is well formed
- $c!\mathtt{Real}.\mathbf{1} \mid c?\mathtt{Int}.\mathbf{1}$ is ill formed

Well-typed participant

Definition

T is *viable* if $T \mid S$ is well formed for some S

Example

$$T \stackrel{\text{def}}{=} c?\mathtt{Int}.\mathbf{1} + c?\mathtt{Bool}.\mathbf{0}$$
$$S \stackrel{\text{def}}{=} c?\mathtt{Int}.\mathbf{0} + c?\mathtt{Bool}.\mathbf{1}$$

- T is viable
- S is viable
- $T \oplus S$ is not viable

Example: global order on channels

$$P \stackrel{\text{def}}{=} a?\llparenthesis x \rrparenthesis.b?\llparenthesis y \rrparenthesis.x!3.x?(z : \texttt{Int}).y!\texttt{true}.0$$

$$P' \stackrel{\text{def}}{=} a?\llparenthesis x \rrparenthesis.b?\llparenthesis y \rrparenthesis.x!3.y!\texttt{true}.x?(z : \texttt{Int}).0$$

$$Q \stackrel{\text{def}}{=} a!(c).b!(d).c?(z : \texttt{Int}).d?(z : \texttt{Bool}).c!5.0$$

$$Q' \stackrel{\text{def}}{=} a!(c).b!(d).c?(z : \texttt{Int}).c!5.d?(z' : \texttt{Bool}).0$$

- deadlock because of cyclic dependency
- $T_P \mid T_Q$ ill-formed (**not viable!**)

Example: global order on channels

$$P \stackrel{\mathrm{def}}{=} a?\llparenthesis x \rrparenthesis.b?\llparenthesis y \rrparenthesis.x!3.x?(z : \mathtt{Int}).y!\mathtt{true}.0$$

$$P' \stackrel{\mathrm{def}}{=} a?\llparenthesis x \rrparenthesis.b?\llparenthesis y \rrparenthesis.x!3.y!\mathtt{true}.x?(z : \mathtt{Int}).0$$

$$Q \stackrel{\mathrm{def}}{=} a!(c).b!(d).c?(z : \mathtt{Int}).d?(z : \mathtt{Bool}).c!5.0$$

$$Q' \stackrel{\mathrm{def}}{=} a!(c).b!(d).c?(z : \mathtt{Int}).c!5.d?(z' : \mathtt{Bool}).0$$

- imposing global order
- $T_P \mid T_{Q'}$ well-formed

Example: global order on channels

$$P \stackrel{\text{def}}{=} a?\llparenthesis x \rrparenthesis.b?\llparenthesis y \rrparenthesis.x!3.x?(z : \mathtt{Int}).y!\mathtt{true}.0$$

$$P' \stackrel{\text{def}}{=} a?\llparenthesis x \rrparenthesis.b?\llparenthesis y \rrparenthesis.x!3.y!\mathtt{true}.x?(z : \mathtt{Int}).0$$

$$Q \stackrel{\text{def}}{=} a!(c).b!(d).c?(z : \mathtt{Int}).d?(z : \mathtt{Bool}).c!5.0$$

$$Q' \stackrel{\text{def}}{=} a!(c).b!(d).c?(z : \mathtt{Int}).c!5.d?(z' : \mathtt{Bool}).0$$

- global order is not necessary
- $T_{P'} \mid T_Q$ well-formed

Example: linearity

$$a?(\!|x|\!).b?(\!|y|\!).x!(\!|y|\!).x?(z : \mathtt{Int}).y!\mathtt{true}.0$$

$$a!(c).b!(d).c?(\!|z|\!).c!5.z?(z' : \mathtt{Bool}).0$$

Subcontract

Definition

$T \preceq S$ if $T \mid R$ well formed implies $S \mid R$ well formed for every R

Examples

- $T \oplus S \preceq T$
- $\pi.T + \pi.S \approx \pi.(T \oplus S)$

  ...very much like the *must* preorder ...
- $\mathbf{0} \preceq T$

$\preceq$ is not a precongruence

$$\mathbf{0} \preceq T$$

Definition (strong subcontract)

Let $\sqsubseteq$ be the largest precongruence included in $\preceq$

Theorem

If T is viable, then $T \preceq S$ iff $T \sqsubseteq S$

- $T \sqsubseteq \mathbf{0}$ iff T is not viable
- if $\mathbf{1} + T \sqsubseteq T$, then T is well formed
- $\pi.\mathbf{0} \sqsubseteq \pi.T$

Outline

On progress

Theorem

If $\vdash P : T$ *and* T *w.f. and* $P \stackrel{\tau}{\Longrightarrow} Q \stackrel{\tau}{\not\longrightarrow}$, *then* Q *has succeeded*

- success = "no pending actions"

On decidability

Proposition

- *well-formedness*
- *viability*
- *subcontract*

are decidable provided that $c!f$ *matches finitely many names*

If a name is sent:

- either it is fresh $c!(a)$
- or it is a public name $c!a$
- or it was received earlier $c?(\mathsf{x}) \cdots d!\mathsf{x}$

Outline

Session types and contracts: a comparison

- optimistic *vs* conservative
- global *vs* compositional

	Session types	Contracts
structuring	++	−−
analysis	−−	++

Concluding remarks

Contributions

1. contracts for processes with channel mobility
2. straightforward solution to global progress (of bounded systems)

Our wish list

- algorithms (almost done)
- choreographic specifications
- expressiveness

Concluding remarks

Contributions

1. contracts for processes with channel mobility
2. straightforward solution to global progress
   (of bounded systems)

Our wish list

- algorithms (almost done)
- choreographic specifications
- expressiveness

Thank you.

Regular does not mean finite-state

- unbounded participants
- unbounded buffers
- state encoded within processes

$$\begin{array}{rcl} P(x : \texttt{Int}) & = & \texttt{deposit?}(y : \texttt{Int}).P(x + y) \\ & & + \texttt{withdraw?}(y : \texttt{Int}).P(\max\{0, x - y\}) \end{array}$$

$$P(0)$$

$$\begin{array}{rcl} P & = & c?(x : \texttt{Int}).(\texttt{deposit?}(y : \texttt{Int}).c!\langle x + y\rangle.P \\ & & + \texttt{withdraw?}(y : \texttt{Int}).c!\langle \max\{0, x - y\}\rangle.P) \\ Q & = & c?(x : \texttt{Int}).c!\langle x\rangle.Q \end{array}$$

$$(\nu c)(P \mid c!\langle 0\rangle.Q)$$

Simulating asynchrony

INPUT

$$\frac{\Gamma \vdash \alpha : \mathtt{Ch} \qquad \Gamma, x : t \vdash P : T}{\Gamma \vdash \alpha?(x : t).P : \alpha?t.T + \alpha?\neg t.\mathbf{0}}$$

C-RECV

$$\frac{\Gamma \vdash \alpha : \mathtt{Ch} \qquad \Gamma, x : \mathtt{Ch} \vdash P : T}{\Gamma \vdash \alpha?(\!|x|\!).P : \alpha?(x).T + \alpha?\neg\mathtt{Ch}.\mathbf{0}}$$