A Theory of Contracts for Web Services

Giuseppe Castagna, Nils Gesbert, Luca Padovani

Université Paris 7, University of Glasgow, Università di Urbino

ParSec meeting 11/12/2007

Web services in a nutshell

- distributed processes
- communicating through standard Web protocols (TCP, HTTP, SOAP)
- exchanging data in platform-neutral format (XML)
- dynamically linked
- with machine-understandable descriptions

Web services in a nutshell

- distributed processes
- communicating through standard Web protocols (TCP, HTTP, SOAP)
- exchanging data in platform-neutral format (XML)
- dynamically linked
- with machine-understandable descriptions

Technologies for Web services

Interface descriptions

- WSDL 1.1 (W3C note, 2001)
- WSDL 2.0 (W3C recommendation, 2007)

Behavioural descriptions

- WSCL 1.0 (W3C note, 2002)
- WSCI 1.0 (W3C note, 2002)
- WS–BPEL 2.0 (OASIS standard, 2007)

"Enabling users to describe business process activities as Web services and define how they can be connected to accomplish specific tasks"

Registries

- UDDI 3.0.2 (OASIS standard, 2004)

"Defining a standard method for enterprises to dynamically discover and invoke Web services"

Technologies for Web services

Interface descriptions

- WSDL 1.1 (W3C note, 2001)
- WSDL 2.0 (W3C recommendation, 2007)

Behavioural descriptions

- WSCL 1.0 (W3C note, 2002)
- WSCI 1.0 (W3C note, 2002)
- WS−BPEL 2.0 (OASIS standard, 2007)

"Enabling users to describe business process activities as Web services and define how they can be connected to accomplish specific tasks"

Registries

- UDDI 3.0.2 (OASIS standard, 2004)

"Defining a standard method for enterprises to dynamically discover and invoke Web services"

Technologies for Web services

Interface descriptions

- `WSDL` 1.1 (W3C note, 2001)
- `WSDL` 2.0 (W3C recommendation, 2007)

Behavioural descriptions

- `WSCL` 1.0 (W3C note, 2002)
- `WSCI` 1.0 (W3C note, 2002)
- `WS−BPEL` 2.0 (OASIS standard, 2007)

"Enabling users to describe business process activities as Web services and define how they can be connected to accomplish specific tasks"

Registries

- `UDDI` 3.0.2 (OASIS standard, 2004)

"Defining a standard method for enterprises to dynamically discover and invoke Web services"

Discovering Web services

Search key

- name
- industrial classification
- location
- ...
- **behavioural type!**

Problem

We need a *semantic* notion of behavioural equivalence which

- preserves client satisfaction
- is abstract (based on the described, observable behaviour)

Plan

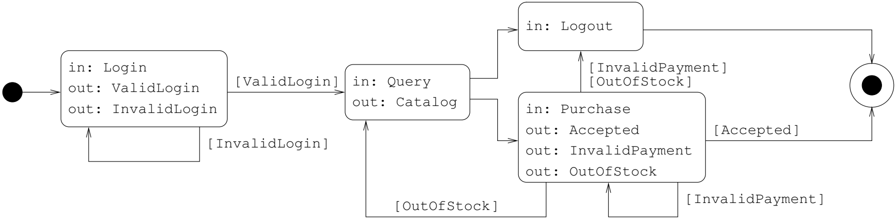
Synthesise *contracts* from Web service descriptions, give contracts a formal semantics, use contracts for searching (and possibly more. . .)

Summary

In this talk. . .

1. understand what contracts look like
2. define client satisfaction (*compliance*)
3. define contract equivalence (*subcontract*)
4. relate compliance and subcontract (*filters*)
5. apply to languages used to implement client/services
6. apply to service discovery

What is a contract?



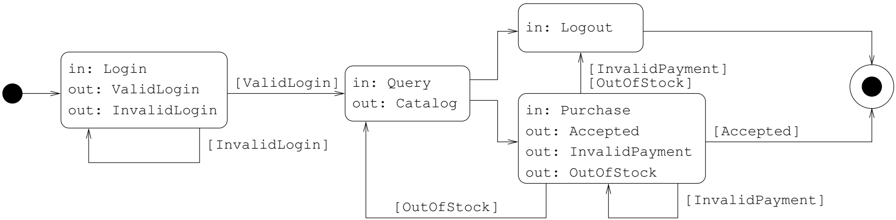
1. Describes sequences of INPUT/OUTPUT actions

   $\texttt{Query}.\overline{\texttt{Catalog}}$

2. $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

What is a contract?



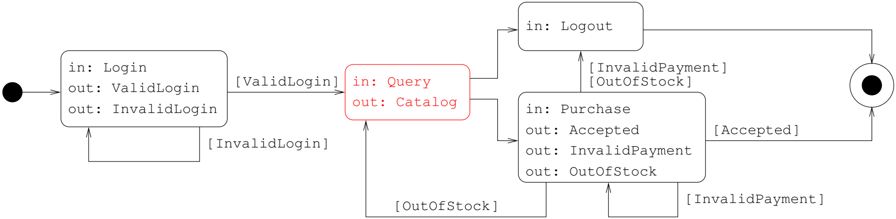
1. Describes sequences of INPUT/OUTPUT actions

   $\texttt{Query}.\overline{\texttt{Catalog}}$

2. $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

What is a contract?



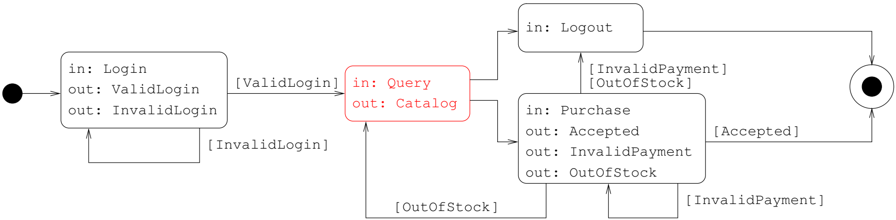
1. Describes sequences of INPUT/OUTPUT actions

   $\texttt{Query}.\overline{\texttt{Catalog}}$

2. $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

What is a contract?



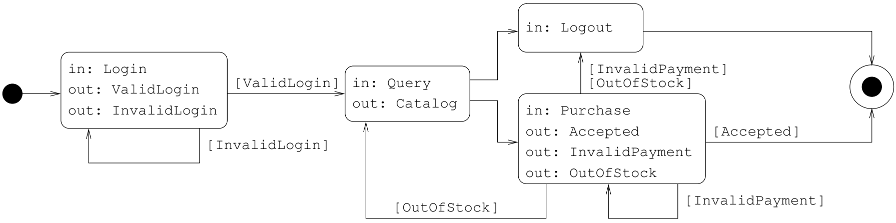
1. Describes sequences of INPUT/OUTPUT actions

   $\texttt{Query}.\overline{\texttt{Catalog}}$

2. $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

What is a contract?



1. Describes sequences of INPUT/OUTPUT actions

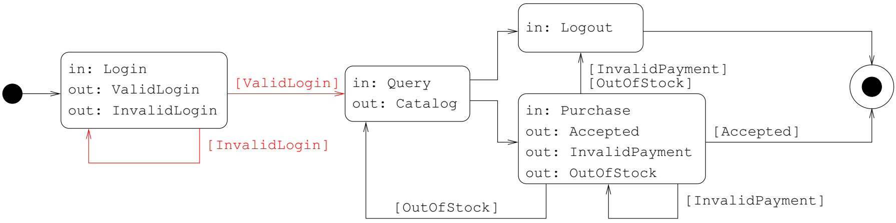
   $\texttt{Query}.\overline{\texttt{Catalog}}$

2. Describes possible internal choices

   $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

What is a contract?



1. Describes sequences of INPUT/OUTPUT actions

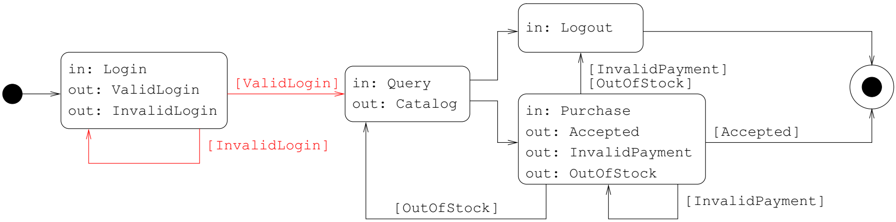
   $\texttt{Query}.\overline{\texttt{Catalog}}$

2. Describes possible internal choices

   $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

What is a contract?



1. Describes sequences of INPUT/OUTPUT actions

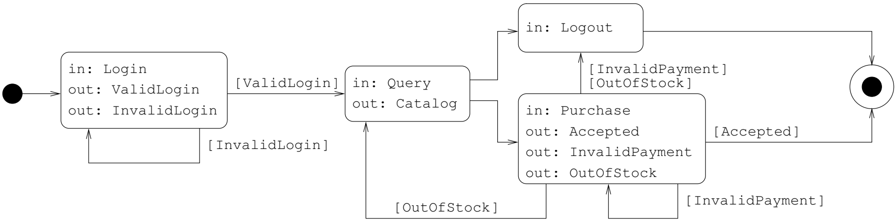
   $\texttt{Query}.\overline{\texttt{Catalog}}$

2. Describes possible internal choices

   $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

What is a contract?



1. Describes sequences of INPUT/OUTPUT actions

   $\mathtt{Query}.\overline{\mathtt{Catalog}}$

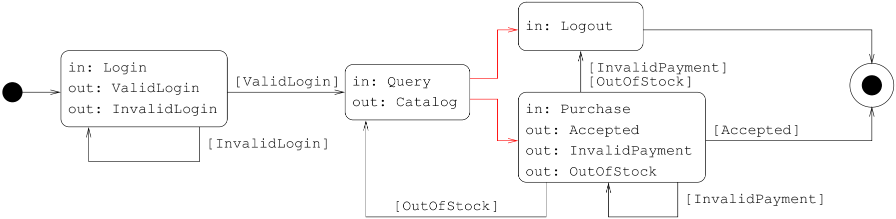
2. Describes possible internal choices

   $\mathtt{Login}.(\overline{\mathtt{ValidLogin}}.\ldots \oplus \overline{\mathtt{InvalidLogin}}.\ldots)$

3. Describes available external choices

   $\mathtt{Query}.\overline{\mathtt{Catalog}}.(\mathtt{Logout}.\ldots + \mathtt{Purchase}.\ldots)$

What is a contract?



1. Describes sequences of INPUT/OUTPUT actions

   $\mathtt{Query}.\overline{\mathtt{Catalog}}$

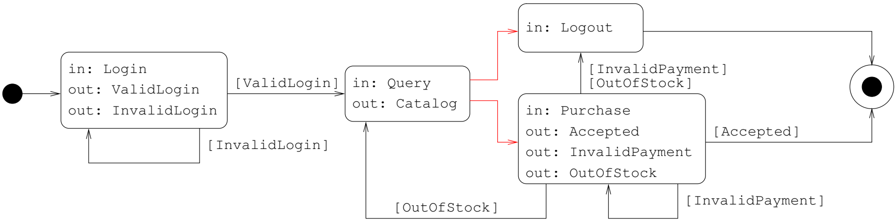
2. Describes possible internal choices

   $\mathtt{Login}.(\overline{\mathtt{ValidLogin}}.\ldots \oplus \overline{\mathtt{InvalidLogin}}.\ldots)$

3. Describes available external choices

   $\mathtt{Query}.\overline{\mathtt{Catalog}}.(\mathtt{Logout}.\ldots + \mathtt{Purchase}.\ldots)$

What is a contract?



1. Describes sequences of INPUT/OUTPUT actions

   $\texttt{Query}.\overline{\texttt{Catalog}}$

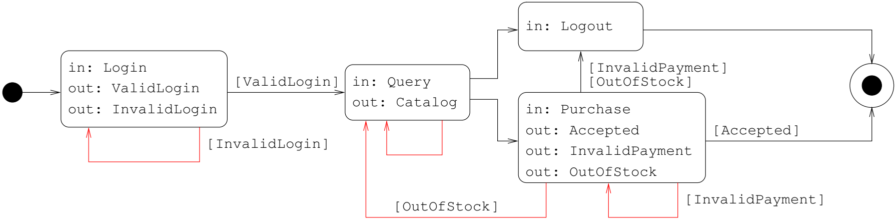
2. Describes possible internal choices

   $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. Describes available external choices

   $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

What is a contract?



1. Describes sequences of INPUT/OUTPUT actions

   $\texttt{Query}.\overline{\texttt{Catalog}}$

2. Describes possible internal choices

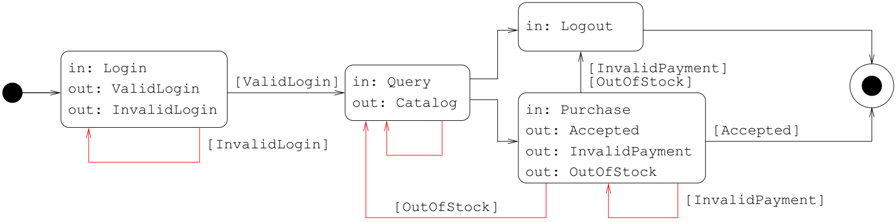
   $\texttt{Login}.(\overline{\texttt{ValidLogin}}.\ldots \oplus \overline{\texttt{InvalidLogin}}.\ldots)$

3. Describes available external choices

   $\texttt{Query}.\overline{\texttt{Catalog}}.(\texttt{Logout}.\ldots + \texttt{Purchase}.\ldots)$

Note that the contract is recursive

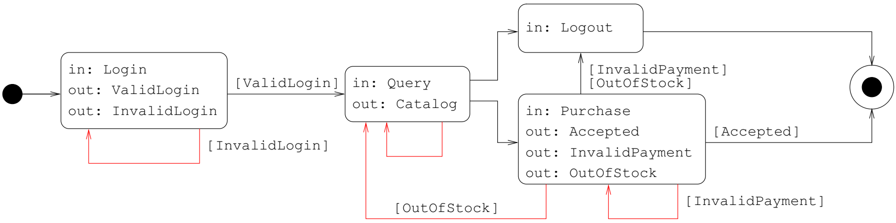
What is a contract?



$$\begin{aligned}&\mathsf{rec}\ x.\mathtt{Login}.(\overline{\mathtt{InvalidLogin}}.x \oplus \overline{\mathtt{ValidLogin}}.\mathsf{rec}\ y.\\ &\quad \mathtt{Query}.\overline{\mathtt{Catalog}}.(y + \mathtt{Logout} + \mathsf{rec}\ z.\mathtt{Purchase}.\\ &\qquad \overline{\mathtt{Accepted}} \oplus \overline{\mathtt{InvalidPayment}}.(z + \mathtt{Logout}) \oplus \overline{\mathtt{OutOfStock}}.(y + \mathtt{Logout})))\end{aligned}$$

Note that the contract is recursive

What is a contract?



$$\text{rec } x.\mathtt{Login}.(\overline{\mathtt{InvalidLogin}}.x \oplus \overline{\mathtt{ValidLogin}}.\text{rec } y.\ \mathtt{Query}.\overline{\mathtt{Catalog}}.(y + \mathtt{Logout} + \text{rec } z.\mathtt{Purchase}.\ \overline{\mathtt{Accepted}} \oplus \overline{\mathtt{InvalidPayment}}.(z + \mathtt{Logout}) \oplus \overline{\mathtt{OutOfStock}}.(y + \mathtt{Logout})))$$

We do not consider recursion in this talk

A formal contract language

contracts σ ::=

$\mathbf{0}$	(*void*)
$\alpha.\sigma$	(*action prefix*)
$\sigma + \sigma$	(*external choice*)
$\sigma \oplus \sigma$	(*internal choice*)

actions α ::=

a	(*receive*)
$\overline{a}$	(*send*)

Names represent types, operations, ...

A formal contract language

contracts	σ	$::=$		
			$\mathbf{0}$	(*void*)
			$\alpha.\sigma$	(*action prefix*)
			$\sigma + \sigma$	(*external choice*)
			$\sigma \oplus \sigma$	(*internal choice*)
actions	α	$::=$		
			a	(*receive*)
			$\overline{a}$	(*send*)

Names represent types, operations, . . .

Two questions:

1. When does a client fit a server of given contract? **(compliance)**
2. When is a contract more general than another? **(subcontracting)**

A formal contract language

contracts	σ	$::=$		
			$\mathbf{0}$	(*void*)
			$\alpha.\sigma$	(*action prefix*)
			$\sigma + \sigma$	(*external choice*)
			$\sigma \oplus \sigma$	(*internal choice*)
actions	α	$::=$		
			a	(*receive*)
			$\overline{a}$	(*send*)

Names represent types, operations, . . .

Two questions:

1. When does a client fit a server of given contract? **(compliance)**
   When it successfully achieves every possible interaction with it
2. When is a contract more general than another? **(subcontracting)**

A formal contract language

contracts σ $::=$

$\mathbf{0}$	(*void*)
$\alpha.\sigma$	(*action prefix*)
$\sigma + \sigma$	(*external choice*)
$\sigma \oplus \sigma$	(*internal choice*)

actions α $::=$

a	(*receive*)
$\overline{a}$	(*send*)

Names represent types, operations, . . .

Two questions:

1. When does a client fit a server of given contract? **(compliance)**
   When it successfully achieves every possible interaction with it
2. When is a contract more general than another? **(subcontracting)**
   When all the clients of the other comply with it

Semantics

The contract of a process describes

1. WHICH actions the process offers

$$\alpha.\sigma \stackrel{\alpha}{\longmapsto} \sigma$$

$$\frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 + \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \not\stackrel{\alpha}{\longmapsto}}{\sigma_1 + \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1'}$$

$$\frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 \oplus \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \not\stackrel{\alpha}{\longmapsto}}{\sigma_1 \oplus \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1'}$$

2. HOW actions are offered

$$\mathbf{0} \Downarrow \emptyset$$

$$\alpha.\sigma \Downarrow \{\alpha\}$$

$$(\sigma + \sigma') \Downarrow \mathrm{R} \cup \mathrm{R}' \qquad \text{if } \sigma \Downarrow \mathrm{R} \text{ and } \sigma' \Downarrow \mathrm{R}'$$

$$(\sigma \oplus \sigma') \Downarrow \mathrm{R} \qquad \text{if either } \sigma \Downarrow \mathrm{R} \text{ or } \sigma' \Downarrow \mathrm{R}$$

Semantics

The contract of a process describes

1. WHICH actions the process offers

$$\alpha.\sigma \stackrel{\alpha}{\longmapsto} \sigma$$

$$\frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 + \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\not\longmapsto}}{\sigma_1 + \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1'}$$

$$\frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 \oplus \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\not\longmapsto}}{\sigma_1 \oplus \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1'}$$

2. HOW actions are offered

$$\mathbf{0} \Downarrow \emptyset$$

$$\alpha.\sigma \Downarrow \{\alpha\}$$

$$(\sigma + \sigma') \Downarrow \mathrm{R} \cup \mathrm{R}' \qquad \text{if } \sigma \Downarrow \mathrm{R} \text{ and } \sigma' \Downarrow \mathrm{R}'$$

$$(\sigma \oplus \sigma') \Downarrow \mathrm{R} \qquad \text{if either } \sigma \Downarrow \mathrm{R} \text{ or } \sigma' \Downarrow \mathrm{R}$$

Semantics

The contract of a process describes

1. WHICH actions the process offers ($\sigma \overset{\alpha}{\longmapsto}$: "$\sigma$ can emit α")

$$\alpha.\sigma \overset{\alpha}{\longmapsto} \sigma$$

$$\frac{\sigma_1 \overset{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \overset{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 + \sigma_2 \overset{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \overset{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \overset{\alpha}{\not\longmapsto}}{\sigma_1 + \sigma_2 \overset{\alpha}{\longmapsto} \sigma_1'}$$

$$\frac{\sigma_1 \overset{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \overset{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 \oplus \sigma_2 \overset{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \overset{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \overset{\alpha}{\not\longmapsto}}{\sigma_1 \oplus \sigma_2 \overset{\alpha}{\longmapsto} \sigma_1'}$$

2. HOW actions are offered

$$\mathbf{0} \Downarrow \emptyset$$

$$\alpha.\sigma \Downarrow \{\alpha\}$$

$$(\sigma + \sigma') \Downarrow \mathrm{R} \cup \mathrm{R}' \qquad \text{if } \sigma \Downarrow \mathrm{R} \text{ and } \sigma' \Downarrow \mathrm{R}'$$

$$(\sigma \oplus \sigma') \Downarrow \mathrm{R} \qquad \text{if either } \sigma \Downarrow \mathrm{R} \text{ or } \sigma' \Downarrow \mathrm{R}$$

Semantics

The contract of a process describes

1. WHICH actions the process offers ($\sigma \overset{\alpha}{\longmapsto}$: "$\sigma$ can emit α")

$$\alpha.\sigma \overset{\alpha}{\longmapsto} \sigma \qquad \frac{\sigma_1 \overset{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \overset{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 + \sigma_2 \overset{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \overset{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \overset{\alpha}{\not\longmapsto}}{\sigma_1 + \sigma_2 \overset{\alpha}{\longmapsto} \sigma_1'}$$

$$\frac{\sigma_1 \overset{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \overset{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 \oplus \sigma_2 \overset{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \overset{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \overset{\alpha}{\not\longmapsto}}{\sigma_1 \oplus \sigma_2 \overset{\alpha}{\longmapsto} \sigma_1'}$$

2. HOW actions are offered

$$\mathbf{0} \Downarrow \emptyset$$

$$\alpha.\sigma \Downarrow \{\alpha\}$$

$$(\sigma + \sigma') \Downarrow \mathrm{R} \cup \mathrm{R}' \qquad \text{if } \sigma \Downarrow \mathrm{R} \text{ and } \sigma' \Downarrow \mathrm{R}'$$

$$(\sigma \oplus \sigma') \Downarrow \mathrm{R} \qquad \text{if either } \sigma \Downarrow \mathrm{R} \text{ or } \sigma' \Downarrow \mathrm{R}$$

Semantics

The contract of a process describes

1. WHICH actions the process offers ($\sigma \stackrel{\alpha}{\longmapsto}$: "$\sigma$ can emit α")

$$\alpha.\sigma \stackrel{\alpha}{\longmapsto} \sigma \qquad \frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 + \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\not\longmapsto}}{\sigma_1 + \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1'}$$

$$\frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_2'}{\sigma_1 \oplus \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1' \oplus \sigma_2'} \qquad \frac{\sigma_1 \stackrel{\alpha}{\longmapsto} \sigma_1' \quad \sigma_2 \stackrel{\alpha}{\not\longmapsto}}{\sigma_1 \oplus \sigma_2 \stackrel{\alpha}{\longmapsto} \sigma_1'}$$

2. HOW actions are offered ($\sigma \Downarrow \mathrm{R}$: "$\sigma$ may offer to choose in R")

$$\begin{array}{ll}
\mathbf{0} \Downarrow \emptyset & \\
\alpha.\sigma \Downarrow \{\alpha\} & \\
(\sigma + \sigma') \Downarrow \mathrm{R} \cup \mathrm{R}' & \text{if } \sigma \Downarrow \mathrm{R} \text{ and } \sigma' \Downarrow \mathrm{R}' \\
(\sigma \oplus \sigma') \Downarrow \mathrm{R} & \text{if either } \sigma \Downarrow \mathrm{R} \text{ or } \sigma' \Downarrow \mathrm{R}
\end{array}$$

Semantics

For instance

$$a \oplus b \overset{a}{\longmapsto} \mathbf{0} \qquad\qquad a + b \overset{a}{\longmapsto} \mathbf{0}$$

$$a \oplus b \overset{b}{\longmapsto} \mathbf{0} \qquad\qquad a + b \overset{b}{\longmapsto} \mathbf{0}$$

but

$$a \oplus b \Downarrow \{a\}$$

$$a \oplus b \Downarrow \{b\}$$

$$a + b \Downarrow \{a, b\}$$

Relating clients and services: **compliance**

A client ρ complies with a service σ ($\rho \dashv \sigma$) if it successfully achieves every possible interaction with the service

- $a.✌ + b.✌ \quad \dashv \quad \overline{a} \oplus \overline{b}$
- $a.✌ + b.✌ \quad \dashv \quad \overline{a}$
- $a.✌ \oplus b.✌ \quad \dashv \quad \overline{a}.c + \overline{b}.\overline{d}$
- $a.✌ \oplus b.✌ \quad \not\dashv \quad \overline{a} \oplus \overline{b}$

(✌ indicates client's satisfaction).

Relating clients and services: **compliance**

A client ρ complies with a service σ ($\rho \dashv \sigma$) if it successfully achieves every possible interaction with the service

- $a.✌ + b.✌ \quad \dashv \quad \overline{a} \oplus \overline{b}$
- $a.✌ + b.✌ \quad \dashv \quad \overline{a}$
- $a.✌ \oplus b.✌ \quad \dashv \quad \overline{a}.c + \overline{b}.\overline{d}$
- $a.✌ \oplus b.✌ \quad \not\dashv \quad \overline{a} \oplus \overline{b}$

(✌ indicates client's satisfaction).

Relating clients and services: **compliance**

A client ρ complies with a service σ ($\rho \dashv \sigma$) if it successfully achieves every possible interaction with the service

- $a.✌ + b.✌ \quad \dashv \quad \overline{a} \oplus \overline{b}$
- $a.✌ + b.✌ \quad \dashv \quad \overline{a}$
- $a.✌ \oplus b.✌ \quad \dashv \quad \overline{a}.c + \overline{b}.\overline{d}$
- $a.✌ \oplus b.✌ \quad \not\dashv \quad \overline{a} \oplus \overline{b}$

(✌ indicates client's satisfaction).

Formally $\rho \dashv \sigma$ iff

1. $\rho \Downarrow \mathrm{R}$, $\sigma \Downarrow \mathrm{S}$, and $\overline{\mathrm{R}} \cap \mathrm{S} = \varnothing$ imply ✌ $\in \mathrm{R}$
2. $\rho \stackrel{\overline{\alpha}}{\longmapsto} \rho'$ and $\sigma \stackrel{\alpha}{\longmapsto} \sigma'$ imply $\rho' \dashv \sigma'$

Relating different services: **subcontract**

Intuition

A client that works with a server σ will also work with a server τ that “does more”: $\sigma \preceq \tau$

1. **Is more deterministic:**

$$\overline{a} \oplus \overline{b}.c \preceq \overline{a}$$

$$\overline{\texttt{InvalidLogin}} \oplus \overline{\texttt{ValidLogin}} \preceq \overline{\texttt{ValidLogin}}$$

2. **Offers more choices:**

$$\overline{a} \preceq \overline{a} + \overline{b}.d$$

$$\texttt{Logout+Purchase} \preceq \texttt{Logout+Purchase+SaveForLater} \quad \text{[width extension]}$$

3. **Offers longer interaction patterns:**

$$\overline{a} \preceq \overline{a}.\overline{b}.d$$

$$\texttt{Purchase}.\overline{\texttt{Accepted}} \preceq \texttt{Purchase}.\overline{\texttt{Accepted}}.\overline{\texttt{Invoice}} \quad \text{[depth extension]}$$

Relating different services: **subcontract**

Intuition

A client that works with a server σ will also work with a server τ that "does more": $\sigma \preceq \tau$

1. **Is more deterministic:**

$$\overline{a} \oplus \overline{b}.c \preceq \overline{a}$$

$\overline{\texttt{InvalidLogin}} \oplus \overline{\texttt{ValidLogin}} \preceq \overline{\texttt{ValidLogin}}$

2. **Offers more choices:**

$$\overline{a} \preceq \overline{a} + \overline{b}.d$$

$\texttt{Logout+Purchase} \preceq \texttt{Logout+Purchase+SaveForLater}$ [width extension]

3. **Offers longer interaction patterns:**

$$\overline{a} \preceq \overline{a}.\overline{b}.d$$

$\texttt{Purchase}.\overline{\texttt{Accepted}} \preceq \texttt{Purchase}.\overline{\texttt{Accepted}}.\overline{\texttt{Invoice}}$ [depth extension]

Relating different services: **subcontract**

Intuition

A client that works with a server σ will also work with a server τ that "does more": $\sigma \preceq \tau$

1. **Is more deterministic:**

$$\overline{a} \oplus \overline{b}.c \preceq \overline{a}$$

$\overline{\texttt{InvalidLogin}} \oplus \overline{\texttt{ValidLogin}} \preceq \overline{\texttt{ValidLogin}}$

2. **Offers more choices:**

$$\overline{a} \preceq \overline{a} + \overline{b}.d$$

$\texttt{Logout+Purchase} \preceq \texttt{Logout+Purchase+SaveForLater}$ [width extension]

3. **Offers longer interaction patterns:**

$$\overline{a} \preceq \overline{a}.\overline{b}.d$$

$\texttt{Purchase}.\overline{\texttt{Accepted}} \preceq \texttt{Purchase}.\overline{\texttt{Accepted}}.\overline{\texttt{Invoice}}$ [depth extension]

Relating different services: **subcontract**

Intuition

A client that works with a server σ will also work with a server τ that "does more": $\sigma \preceq \tau$

1. **Is more deterministic:**

$$\overline{a} \oplus \overline{b}.c \;\preceq\; \overline{a}$$

$\overline{\texttt{InvalidLogin}} \oplus \overline{\texttt{ValidLogin}} \preceq \overline{\texttt{ValidLogin}}$

2. **Offers more choices:**

$$\overline{a} \;\preceq\; \overline{a} + \overline{b}.d$$

$\texttt{Logout+Purchase} \preceq \texttt{Logout+Purchase+SaveForLater}$ [width extension]

3. **Offers longer interaction patterns:**

$$\overline{a} \;\preceq\; \overline{a}.\overline{b}.d$$

$\texttt{Purchase}.\overline{\texttt{Accepted}} \preceq \texttt{Purchase}.\overline{\texttt{Accepted}}.\overline{\texttt{Invoice}}$ [depth extension]

Relating different services: **subcontract**

Intuition

A client that works with a server σ will also work with a server τ that "does more": $\sigma \preceq \tau$

1. **Is more deterministic:**

$$\overline{a} \oplus \overline{b}.c \preceq \overline{a}$$

$\overline{\texttt{InvalidLogin}} \oplus \overline{\texttt{ValidLogin}} \preceq \overline{\texttt{ValidLogin}}$

2. **Offers more choices:**

$$\overline{a} \preceq \overline{a} + \overline{b}.d$$

$\texttt{Logout}+\texttt{Purchase} \preceq \texttt{Logout}+\texttt{Purchase}+\texttt{SaveForLater}$ [width extension]

3. **Offers longer interaction patterns:**

$$\overline{a} \preceq \overline{a}.\overline{b}.d$$

$\texttt{Purchase}.\overline{\texttt{Accepted}} \preceq \texttt{Purchase}.\overline{\texttt{Accepted}}.\overline{\texttt{Invoice}}$ [depth extension]

Relating different services: **subcontract**

Intuition

A client that works with a server σ will also work with a server τ that "does more": $\sigma \preceq \tau$

1. **Is more deterministic:**

$$\overline{a} \oplus \overline{b}.c \preceq \overline{a}$$

$\overline{\texttt{InvalidLogin}} \oplus \overline{\texttt{ValidLogin}} \preceq \overline{\texttt{ValidLogin}}$

2. **Offers more choices:**

$$\overline{a} \preceq \overline{a} + \overline{b}.d$$

$\texttt{Logout}+\texttt{Purchase} \preceq \texttt{Logout}+\texttt{Purchase}+\texttt{SaveForLater}$ [width extension]

3. **Offers longer interaction patterns:**

$$\overline{a} \preceq \overline{a}.\overline{b}.d$$

$\texttt{Purchase}.\overline{\texttt{Accepted}} \preceq \texttt{Purchase}.\overline{\texttt{Accepted}}.\overline{\texttt{Invoice}}$ [depth extension]

Relating different services: **subcontract**

Intuition

A client that works with a server σ will also work with a server τ that "does more": $\sigma \preceq \tau$

1. **Is more deterministic:**

$$\overline{a} \oplus \overline{b}.c \;\preceq\; \overline{a}$$

$\overline{\texttt{InvalidLogin}} \oplus \overline{\texttt{ValidLogin}} \preceq \overline{\texttt{ValidLogin}}$

2. **Offers more choices:**

$$\overline{a} \;\preceq\; \overline{a} + \overline{b}.d$$

$\texttt{Logout}+\texttt{Purchase} \preceq \texttt{Logout}+\texttt{Purchase}+\texttt{SaveForLater}$ [width extension]

3. **Offers longer interaction patterns:**

$$\overline{a} \;\preceq\; \overline{a}.\overline{b}.d$$

$\texttt{Purchase}.\overline{\texttt{Accepted}} \preceq \texttt{Purchase}.\overline{\texttt{Accepted}}.\overline{\texttt{Invoice}}$ [depth extension]

Relating different services: **subcontract**

Intuition

A client that works with a server σ will also work with a server τ that "does more": $\sigma \preceq \tau$

1. **Is more deterministic:**

$$\overline{a} \oplus \overline{b}.c \preceq \overline{a}$$

$\overline{\texttt{InvalidLogin}} \oplus \overline{\texttt{ValidLogin}} \preceq \overline{\texttt{ValidLogin}}$

2. **Offers more choices:**

$$\overline{a} \preceq \overline{a} + \overline{b}.d$$

$\texttt{Logout}+\texttt{Purchase} \preceq \texttt{Logout}+\texttt{Purchase}+\texttt{SaveForLater}$ [width extension]

3. **Offers longer interaction patterns:**

$$\overline{a} \preceq \overline{a}.\overline{b}.d$$

$\texttt{Purchase}.\overline{\texttt{Accepted}} \preceq \texttt{Purchase}.\overline{\texttt{Accepted}}.\overline{\texttt{Invoice}}$ [depth extension]

Is "$\preceq$" an (inverse) subtyping relation? Apparent mismatch in 1 and 2

Compliance and Subcontract **mismatch**

- Note:

$$\overline{a} \oplus \overline{b}.c \overset{❶}{\preceq} \overline{a} \overset{❷}{\preceq} \overline{a} + \overline{b}.d$$

- but for a client $a.\text{✌} + b.\overline{c}.\text{✌}$:

$$a.\text{✌} + b.\overline{c}.\text{✌} \dashv \overline{a} \oplus \overline{b}.c \qquad a.\text{✌} + b.\overline{c}.\text{✌} \not\dashv \overline{a} + \overline{b}.d$$

- Can we replace a server $\overline{a} \oplus \overline{b}.c$ for a $\overline{a} + \overline{b}.d$ one (i.e. $\overline{a} \oplus \overline{b}.c\ :>\ \overline{a} + \overline{b}.d$)?

- YES if ":>" uses explicit coercions (rather than implicit ones)

Filter-out foreign actions
(e.g. $\overline{b}$ and d for ❷)

Compliance and Subcontract **mismatch**

- Note:

$$\overline{a} \oplus \overline{b}.c \quad \overset{1}{\preceq} \quad \overline{a} \quad \overset{2}{\preceq} \quad \overline{a} + \overline{b}.d$$

- but for a client $a.\text{✌} + b.\overline{c}.\text{✌}$:

$$a.\text{✌} + b.\overline{c}.\text{✌} \dashv \overline{a} \oplus \overline{b}.c \qquad a.\text{✌} + b.\overline{c}.\text{✌} \not\dashv \overline{a} + \overline{b}.d$$

- Can we replace a server $\overline{a} \oplus \overline{b}.c$ for a $\overline{a} + \overline{b}.d$ one (i.e. $\overline{a} \oplus \overline{b}.c \ :> \overline{a} + \overline{b}.d$)?

- YES if ":>" uses explicit coercions (rather than implicit ones)

Filter-out foreign actions
(e.g. $\overline{b}$ and d for 2)

Compliance and Subcontract **mismatch**

- Note:

$$\overline{a} \oplus \overline{b}.c \overset{❶}{\preceq} \overline{a} \overset{❷}{\preceq} \overline{a} + \overline{b}.d$$

- but for a client $a.✌ + b.\overline{c}.✌$:

$$a.✌ + b.\overline{c}.✌ \dashv \overline{a} \oplus \overline{b}.c \qquad a.✌ + b.\overline{c}.✌ \not\dashv \overline{a} + \overline{b}.d$$

- Can we replace a server $\overline{a} \oplus \overline{b}.c$ for a $\overline{a} + \overline{b}.d$ one (i.e. $\overline{a} \oplus \overline{b}.c$:> $\overline{a} + \overline{b}.d$)?

- YES if ":>" uses explicit coercions (rather than implicit ones)

Filter-out foreign actions
(e.g. $\overline{b}$ and d for ❷)

Compliance and Subcontract **mismatch**

- Note:

$$\overline{a} \oplus \overline{b}.c \quad \overset{❶}{\preceq} \quad \overline{a} \quad \overset{❷}{\preceq} \quad \overline{a} + \overline{b}.d$$

- but for a client $a.✌ + b.\overline{c}.✌$:

$$a.✌ + b.\overline{c}.✌ \dashv \overline{a} \oplus \overline{b}.c \qquad a.✌ + b.\overline{c}.✌ \not\dashv \overline{a} + \overline{b}.d$$

- Can we replace a server $\overline{a} \oplus \overline{b}.c$ for a $\overline{a} + \overline{b}.d$ one (i.e. $\overline{a} \oplus \overline{b}.c \;:>\; \overline{a} + \overline{b}.d$)?

- YES if ":>" uses explicit coercions (rather than implicit ones)

Filter-out foreign actions
(e.g. $\overline{b}$ and d for ❷)

Compliance and Subcontract **mismatch**

- Note:

$$\overline{a} \oplus \overline{b}.c \quad \overset{❶}{\preceq} \quad \overline{a} \quad \overset{❷}{\preceq} \quad \overline{a} + \overline{b}.d$$

- but for a client $a.✌ + b.\overline{c}.✌$:

$$a.✌ + b.\overline{c}.✌ \dashv \overline{a} \oplus \overline{b}.c \qquad a.✌ + b.\overline{c}.✌ \not\dashv \overline{a} + \overline{b}.d$$

- Can we replace a server $\overline{a} \oplus \overline{b}.c$ for a $\overline{a} + \overline{b}.d$ one (i.e. $\overline{a} \oplus \overline{b}.c$:> $\overline{a} + \overline{b}.d$)?

- YES if ":>" uses explicit coercions (rather than implicit ones)

Filter-out foreign actions
(e.g. $\overline{b}$ and d for ❷)

Gluing compliance and subcontracting: **Filters**

filters $\qquad f ::= \coprod_{\alpha \in \mathrm{A}} \alpha.f_\alpha$

Transition relation of filters

$$\coprod_{\alpha \in \mathrm{A}} \alpha.f_\alpha \stackrel{\beta}{\longmapsto} f_\beta \qquad \text{if } \beta \in \mathrm{A}$$

Contract coercion through a filter

$$\begin{array}{rcll} f(\mathbf{0}) & = & \mathbf{0} & \\ f(\alpha.\sigma) & = & \mathbf{0} & \text{if } f \stackrel{\alpha}{\not\longmapsto} \\ f(\alpha.\sigma) & = & \alpha.f'(\sigma) & \text{if } f \stackrel{\alpha}{\longmapsto} f' \\ f(\sigma_1 + \sigma_2) & = & f(\sigma_1) + f(\sigma_2) & \\ f(\sigma_1 \oplus \sigma_2) & = & f(\sigma_1) \oplus f(\sigma_2) & \end{array}$$

Property

$$\sigma \preceq \tau \quad \wedge \quad \rho \dashv \sigma \quad \Longleftrightarrow \quad \sigma \dashv f(\tau) \quad \text{for some filter } f$$

Gluing compliance and subcontracting: **Filters**

filters $\quad f ::= \coprod_{\alpha \in \mathrm{A}} \alpha.f_\alpha$

Transition relation of filters

$$\coprod_{\alpha \in \mathrm{A}} \alpha.f_\alpha \xmapsto{\beta} f_\beta \qquad \text{if } \beta \in \mathrm{A}$$

Contract coercion through a filter

$$\begin{array}{rcll} f(\mathbf{0}) & = & \mathbf{0} & \\ f(\alpha.\sigma) & = & \mathbf{0} & \text{if } f \not\xmapsto{\alpha} \\ f(\alpha.\sigma) & = & \alpha.f'(\sigma) & \text{if } f \xmapsto{\alpha} f' \\ f(\sigma_1 + \sigma_2) & = & f(\sigma_1) + f(\sigma_2) & \\ f(\sigma_1 \oplus \sigma_2) & = & f(\sigma_1) \oplus f(\sigma_2) & \end{array}$$

Property

$$\sigma \preceq \tau \quad \wedge \quad \rho \dashv \sigma \quad \iff \quad \sigma \dashv f(\tau) \quad \text{for some filter } f$$

Gluing compliance and subcontracting: **Filters**

filters $\quad f ::= \coprod_{\alpha \in \mathrm{A}} \alpha.f_\alpha$

Transition relation of filters

$$\coprod_{\alpha \in \mathrm{A}} \alpha.f_\alpha \overset{\beta}{\longmapsto} f_\beta \qquad \text{if } \beta \in \mathrm{A}$$

Contract coercion through a filter

$$\begin{array}{rcll} f(\mathbf{0}) & = & \mathbf{0} & \\ f(\alpha.\sigma) & = & \mathbf{0} & \text{if } f \overset{\alpha}{\not\longmapsto} \\ f(\alpha.\sigma) & = & \alpha.f'(\sigma) & \text{if } f \overset{\alpha}{\longmapsto} f' \\ f(\sigma_1 + \sigma_2) & = & f(\sigma_1) + f(\sigma_2) & \\ f(\sigma_1 \oplus \sigma_2) & = & f(\sigma_1) \oplus f(\sigma_2) & \end{array}$$

Property

$$\sigma \preceq \tau \quad \wedge \quad \rho \dashv \sigma \quad \Longleftrightarrow \quad \sigma \dashv f(\tau) \quad \text{for some filter } f$$

Gluing compliance and subcontracting: **Filters**

$$\textbf{filters} \qquad f ::= \coprod_{\alpha \in \mathrm{A}} \alpha.f_\alpha$$

Transition relation of filters

$$\coprod_{\alpha \in \mathrm{A}} \alpha.f_\alpha \stackrel{\beta}{\longmapsto} f_\beta \qquad \text{if } \beta \in \mathrm{A}$$

Contract coercion through a filter

$$\begin{array}{rcll} f(\mathbf{0}) & = & \mathbf{0} & \\ f(\alpha.\sigma) & = & \mathbf{0} & \text{if } f \stackrel{\alpha}{\not\longmapsto} \\ f(\alpha.\sigma) & = & \alpha.f'(\sigma) & \text{if } f \stackrel{\alpha}{\longmapsto} f' \\ f(\sigma_1 + \sigma_2) & = & f(\sigma_1) + f(\sigma_2) & \\ f(\sigma_1 \oplus \sigma_2) & = & f(\sigma_1) \oplus f(\sigma_2) & \end{array}$$

Property

$$\sigma \preceq \tau \quad \wedge \quad \rho \dashv \sigma \quad \Longleftrightarrow \quad \sigma \dashv f(\tau) \quad \text{for some filter } f$$

Results

Filters are “proofs” of subcontracting

$$f : \sigma \preceq \tau$$

- deduction system for subcontracting: $f : \sigma \preceq \tau$,
- algebraic theory for filters
- existence and effectiveness of a most general filter
  (via cut-elimination, yields subcontracting coherence)
- subcontracting decidability

Compliance characterises *must testing*

$$\sigma \sqsubseteq_{must} \tau \iff \rho \dashv \sigma \text{ implies } \rho \dashv \tau$$

Corollary

$$\sigma \preceq \tau \iff \rho \sqsubseteq_{must} f(\tau) \quad \text{for some filter } f$$

Results

Filters are "proofs" of subcontracting

$$f : \sigma \preceq \tau$$

- deduction system for subcontracting: $f : \sigma \preceq \tau$,
- algebraic theory for filters
- existence and effectiveness of a most general filter (via cut-elimination, yields subcontracting coherence)
- subcontracting decidability

Compliance characterises *must testing*

$$\sigma \sqsubseteq_{must} \tau \iff \rho \dashv \sigma \text{ implies } \rho \dashv \tau$$

Corollary

$$\sigma \preceq \tau \iff \rho \sqsubseteq_{must} f(\tau) \quad \text{for some filter } f$$

Results

Filters are "proofs" of subcontracting

$$f : \sigma \preceq \tau$$

- deduction system for subcontracting: $f : \sigma \preceq \tau$,
- algebraic theory for filters
- existence and effectiveness of a most general filter (via cut-elimination, yields subcontracting coherence)
- subcontracting decidability

Compliance characterises *must testing*

$$\sigma \sqsubseteq_{must} \tau \iff \rho \dashv \sigma \text{ implies } \rho \dashv \tau$$

Corollary

$$\sigma \preceq \tau \iff \rho \sqsubseteq_{must} f(\tau) \quad \text{for some filter } f$$

Results

Filters are "proofs" of subcontracting

$$f : \sigma \preceq \tau$$

- deduction system for subcontracting: $f : \sigma \preceq \tau$,
- algebraic theory for filters
- existence and effectiveness of a most general filter (via cut-elimination, yields subcontracting coherence)
- subcontracting decidability

Compliance characterises *must testing*

$$\sigma \sqsubseteq_{must} \tau \quad \Longleftrightarrow \quad \rho \dashv \sigma \text{ implies } \rho \dashv \tau$$

Corollary

$$\sigma \preceq \tau \quad \Longleftrightarrow \quad \rho \sqsubseteq_{must} f(\tau) \quad \text{for some filter } f$$

Results

Filters are "proofs" of subcontracting

$$f : \sigma \preceq \tau$$

- deduction system for subcontracting: $f : \sigma \preceq \tau$,
- algebraic theory for filters
- existence and effectiveness of a most general filter
  (via cut-elimination, yields subcontracting coherence)
- subcontracting decidability

Compliance characterises *must testing*

$$\sigma \sqsubseteq_{must} \tau \iff \rho \dashv \sigma \text{ implies } \rho \dashv \tau$$

Corollary

$$\sigma \preceq \tau \iff \rho \sqsubseteq_{must} f(\tau) \quad \text{for some filter } f$$

Results

Filters are "proofs" of subcontracting

$$f : \sigma \preceq \tau$$

- deduction system for subcontracting: $f : \sigma \preceq \tau$,
- algebraic theory for filters
- existence and effectiveness of a most general filter
  (via cut-elimination, yields subcontracting coherence)
- subcontracting decidability

Compliance characterises *must testing*

$$\sigma \sqsubseteq_{must} \tau \iff \rho \dashv \sigma \text{ implies } \rho \dashv \tau$$

Corollary

$$\sigma \preceq \tau \iff \rho \sqsubseteq_{must} f(\tau) \quad \text{for some filter } f$$

Results

Filters are “proofs” of subcontracting

$$f : \sigma \preceq \tau$$

- deduction system for subcontracting: $f : \sigma \preceq \tau$,
- algebraic theory for filters
- existence and effectiveness of a most general filter
  (via cut-elimination, yields subcontracting coherence)
- subcontracting decidability

Compliance characterises *must testing*

$$\sigma \sqsubseteq_{must} \tau \quad \iff \quad \rho \dashv \sigma \text{ implies } \rho \dashv \tau$$

Corollary

$$\sigma \preceq \tau \quad \iff \quad \rho \sqsubseteq_{must} f(\tau) \quad \text{for some filter } f$$

Some details

- Identity:

$$I_\sigma \quad \stackrel{\text{def}}{=} \quad \coprod_{\sigma \stackrel{\alpha}{\longmapsto} \sigma'} \alpha . I_{\sigma'}$$

- Intersection: composes two filters

$$f \wedge g \quad \stackrel{\text{def}}{=} \quad \coprod_{\alpha \in \mathrm{A} \cap \mathrm{B}} \alpha . (f_\alpha \wedge g_\alpha)$$

- Union: merges two filters

$$f \vee g \quad \stackrel{\text{def}}{=} \quad \coprod_{\alpha \in \mathrm{A} \cup \mathrm{B}} \alpha . \begin{cases} f_\alpha \vee g_\alpha, & \alpha \in \mathrm{A} \cap \mathrm{B} \\ f_\alpha, & \alpha \in \mathrm{A} \setminus \mathrm{B} \\ g_\alpha, & \alpha \in \mathrm{B} \setminus \mathrm{A} \end{cases}$$

$$(\sigma = \tau \quad \stackrel{def}{\equiv} \quad I_\sigma : \sigma \leqslant \tau \text{ and } I_\tau : \tau \leqslant \sigma)$$

$$\begin{array}{rl}
\sigma + \sigma = \sigma & \sigma \oplus \sigma = \sigma \\
\sigma + \tau = \tau + \sigma & \sigma \oplus \tau = \tau \oplus \sigma \\
\sigma + (\sigma' + \sigma'') = (\sigma + \sigma') + \sigma'' & \sigma \oplus (\sigma' \oplus \sigma'') = (\sigma \oplus \sigma') \oplus \sigma'' \\
\sigma + (\sigma' \oplus \sigma'') = (\sigma + \sigma') \oplus (\sigma + \sigma'') & \sigma \oplus (\sigma' + \sigma'') = (\sigma \oplus \sigma') + (\sigma \oplus \sigma'')
\end{array}$$

$$\sigma + \mathbf{0} = \sigma \qquad \alpha.\sigma + \alpha.\tau = \alpha.(\sigma \oplus \tau) \qquad \alpha.\sigma \oplus \alpha.\tau = \alpha.(\sigma \oplus \tau)$$

(Must)
$$I_\sigma : \sigma \oplus \tau \leq \sigma$$

(DepthExt)
$$\mathbf{0} : \mathbf{0} \leq \sigma$$

(Weakening)
$$\frac{f : \sigma \leq \tau \quad g \wedge I_\tau \leqslant f}{f \vee g : \sigma \leq \tau}$$

(Transitivity)
$$\frac{f : \sigma \leq \sigma' \quad g : \sigma' \leq \sigma''}{f \wedge g : \sigma \leq \sigma''}$$

(Prefix)
$$\frac{f : \sigma \leq \tau}{\alpha.f : \alpha.\sigma \leq \alpha.\tau}$$

(IntChoice)
$$\frac{f : \sigma \leq \sigma' \quad f : \tau \leq \tau'}{f : \sigma \oplus \tau \leq \sigma' \oplus \tau'}$$

(ExtChoice)
$$\frac{f : \sigma \leq \sigma' \quad f : \tau \leq \tau'}{f : \sigma + \tau \leq \sigma' + \tau'}$$

Some properties

- The sought subtyping relations

(Must) $\mathbb{I}_\sigma : \sigma \oplus \tau \leq \sigma$

(DepExt+Prefix) $\alpha.\mathbf{0} : \alpha.\mathbf{0} \leq \alpha.\sigma$

(DepExt+ExtCh+Weak)
$$\frac{\mathbb{I}_\sigma \wedge \mathbb{I}_\tau \leqslant \mathbf{0}}{\mathbb{I}_\sigma : \sigma \leq \sigma + \tau}$$

- Canonical filter for $\sigma \leq \tau$ (algorithmic)

$$\bigvee_{\{f \,\mid\, f:\sigma\leq\tau \text{ and } f\leqslant \mathbb{I}_\tau\}} f$$

- Characterisation of must testing for canonical filters

$$\sigma \sqsubseteq_{\mathtt{must}} \tau \iff \mathbb{I}_\tau : \sigma \leq \tau$$

Application to languages

Our contracts work for any language as long as they come equipped with:

1. **An LTS**

$$P \xrightarrow{\mu} P'$$

μ is either a visible action or an invisible τ action

2. **A type system**

$$\vdash P : \sigma$$

σ is a contract

3. **The latter abstracts the former:**
   1. If $\vdash P : \sigma$ and $\sigma \overset{\alpha}{\longmapsto}$, then $P \xrightarrow{\alpha}$
   2. If $\vdash P : \sigma$ and $P \xrightarrow{\mu} P'$ then $\vdash P' : \sigma'$ and
      - if $\mu = \tau$, then $\sigma \sqsubseteq_{must} \sigma'$
      - if $\mu = \alpha$, then $\sigma \overset{\alpha}{\longmapsto} \sqsubseteq_{must} \sigma'$

$\sqsubseteq_{must}$ measures non-determinism

Application to languages

Our contracts work for any language as long as they come equipped with:

1. **An LTS**

$$P \xrightarrow{\mu} P'$$

μ is either a visible action or an invisible τ action

2. **A type system**

$$\vdash P : \sigma$$

σ is a contract

3. **The latter abstracts the former:**
   1. If $\vdash P : \sigma$ and $\sigma \overset{\alpha}{\longmapsto}$, then $P \xrightarrow{\alpha}$
   2. If $\vdash P : \sigma$ and $P \xrightarrow{\mu} P'$ then $\vdash P' : \sigma'$ and
      - if $\mu = \tau$, then $\sigma \sqsubseteq_{must} \sigma'$
      - if $\mu = \alpha$, then $\sigma \overset{\alpha}{\longmapsto} \sqsubseteq_{must} \sigma'$

$\sqsubseteq_{must}$ measures non-determinism

Application to languages

Our contracts work for any language as long as they come equipped with:

1. **An LTS**

$$P \xrightarrow{\mu} P'$$

μ is either a visible action or an invisible τ action

2. **A type system**

$$\vdash P : \sigma$$

σ is a contract

3. **The latter abstracts the former:**
   1. If $\vdash P : \sigma$ and $\sigma \overset{\alpha}{\longmapsto}$, then $P \xrightarrow{\alpha}$
   2. If $\vdash P : \sigma$ and $P \xrightarrow{\mu} P'$ then $\vdash P' : \sigma'$ and
      - if $\mu = \tau$, then $\sigma \sqsubseteq_{must} \sigma'$
      - if $\mu = \alpha$, then $\sigma \overset{\alpha}{\longmapsto} \sqsubseteq_{must} \sigma'$

$\sqsubseteq_{must}$ measures non-determinism

Application to languages

Our contracts work for any language as long as they come equipped with:

1. **An LTS**

$$P \xrightarrow{\mu} P'$$

μ is either a visible action or an invisible $\boldsymbol{\tau}$ action

2. **A type system**

$$\vdash P : \sigma$$

σ is a contract

3. **The latter abstracts the former:**
   1. If $\vdash P : \sigma$ and $\sigma \overset{\alpha}{\longmapsto}$, then $P \xrightarrow{\alpha}$
   2. If $\vdash P : \sigma$ and $P \xrightarrow{\mu} P'$ then $\vdash P' : \sigma'$ and
      - if $\mu = \boldsymbol{\tau}$, then $\sigma \sqsubseteq_{must} \sigma'$
      - if $\mu = \alpha$, then $\sigma \overset{\alpha}{\longmapsto} \sqsubseteq_{must} \sigma'$

$\sqsubseteq_{must}$ measures non-determinism

Process compliance

By the LTS we define sessions and compliance *for processes*

- If $P \xrightarrow{\alpha} P'$ and $Q \xrightarrow{\overline{\alpha}} Q'$ then $P \parallel Q \longrightarrow P' \parallel Q'$ (plus τ-moves)
- Client P complies with server Q (noted $P \dashv Q$) if
  - if $P \xrightarrow{\mu}$, then $\mu = ✌$ or
  - $P \parallel Q \longrightarrow P' \parallel Q'$ and $P' \dashv Q'$

Theorem (Process compliance)

If $\vdash P : \rho$ and $\vdash Q : \sigma$ and $\rho \dashv \sigma$, then $P \dashv Q$

Process compliance

By the LTS we define sessions and compliance *for processes*

- If $P \xrightarrow{\alpha} P'$ and $Q \xrightarrow{\overline{\alpha}} Q'$ then $P \parallel Q \longrightarrow P' \parallel Q'$ (plus τ-moves)
- Client P complies with server Q (noted $P \dashv Q$) if
  - if $P \xrightarrow{\mu}$, then $\mu = ✌$ or
  - $P \parallel Q \longrightarrow P' \parallel Q'$ and $P' \dashv Q'$

Theorem (Process compliance)

If $\vdash P : \rho$ *and* $\vdash Q : \sigma$ *and* $\rho \dashv \sigma$, *then* $P \dashv Q$

Process compliance

By the LTS we define sessions and compliance *for processes*

- If $P \xrightarrow{\alpha} P'$ and $Q \xrightarrow{\overline{\alpha}} Q'$ then $P \parallel Q \longrightarrow P' \parallel Q'$ (plus τ-moves)
- Client P complies with server Q (noted $P \dashv Q$) if
  - if $P \xrightarrow{\mu}$, then $\mu = ✌$ or
  - $P \parallel Q \longrightarrow P' \parallel Q'$ and $P' \dashv Q'$

Theorem (Process compliance)

If $\vdash P : \rho$ and $\vdash Q : \sigma$ and $\rho \dashv \sigma$, then $P \dashv Q$

Process Filtering

Add filters to the language: $f[P]$

Transition rules for filters

$$\text{(Filter1)}\quad \frac{P \xrightarrow{\alpha} P' \qquad f \xmapsto{\alpha} f'}{f[P] \xrightarrow{\alpha} f'[P']} \qquad\qquad \text{(Filter2)}\quad \frac{P \xrightarrow{\tau} P'}{f[P] \xrightarrow{\tau} f[P']}$$

Typing rules for filters

$$\text{(T-Filter)}\quad \frac{\vdash P : \sigma}{\vdash f[P] : f(\sigma)}$$

"Subject reduction" still holds

Theorem (Process filtering)

Process Filtering

Add filters to the language: $f[P]$

Transition rules for filters

$$\text{(Filter1)}\quad \frac{P \xrightarrow{\alpha} P' \qquad f \xmapsto{\alpha} f'}{f[P] \xrightarrow{\alpha} f'[P']} \qquad\qquad \text{(Filter2)}\quad \frac{P \xrightarrow{\tau} P'}{f[P] \xrightarrow{\tau} f[P']}$$

Typing rules for filters

$$\text{(T-Filter)}\quad \frac{\vdash P : \sigma}{\vdash f[P] : f(\sigma)}$$

“Subject reduction” still holds

Theorem (Process filtering)

Let $\vdash P : \rho$ with $\rho \dashv \sigma$. If $\vdash Q : \tau$ and $f : \sigma \preceq \tau$, then $P \dashv f[Q]$

Process Filtering

Add filters to the language: $f[P]$

Transition rules for filters

$$\text{(FILTER1)}\ \frac{P \xrightarrow{\alpha} P' \qquad f \xmapsto{\alpha} f'}{f[P] \xrightarrow{\alpha} f'[P']} \qquad \text{(FILTER2)}\ \frac{P \xrightarrow{\tau} P'}{f[P] \xrightarrow{\tau} f[P']}$$

Typing rules for filters

$$\text{(T-FILTER)}\ \frac{\vdash P : \sigma}{\vdash f[P] : f(\sigma)}$$

"Subject reduction" still holds

Theorem (Process filtering)

Let $\vdash P : \rho$ with $\rho \dashv \sigma$. If $\vdash Q : \tau$ and $f : \sigma \preceq \tau$, then $P \dashv f[Q]$

Process Filtering

Add filters to the language: $f[P]$

Transition rules for filters

$$\text{(FILTER1)}\quad \frac{P \xrightarrow{\alpha} P' \qquad f \xmapsto{\alpha} f'}{f[P] \xrightarrow{\alpha} f'[P']} \qquad\qquad \text{(FILTER2)}\quad \frac{P \xrightarrow{\tau} P'}{f[P] \xrightarrow{\tau} f[P']}$$

Typing rules for filters

$$\text{(T-FILTER)}\quad \frac{\vdash P : \sigma}{\vdash f[P] : f(\sigma)}$$

"Subject reduction" still holds

Theorem (Process filtering)

Let $\vdash P : \rho$ with $\rho \dashv \sigma$. If $\vdash Q : \tau$ and $f : \sigma \preceq \tau$, then $P \dashv f[Q]$

Application to service discovery

Search for services compliant with a client ρ:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid \rho \dashv f(\sigma)\}$$

Call one of the services in the result by using the associated filter.

More efficient using subcontracts and caching

1. Compute $\rho^{\perp}$ the "canonical server" of ρ
2. Return all compatible servers:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma\} \qquad \text{(cached)}$$

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

$$\texttt{discover}(\rho, g) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma \ \wedge g \leq f\}$$

E.g.: provide at least $\texttt{Login}.\overline{\texttt{ValidLogin}}.\texttt{Query}.\overline{\texttt{Catalog}}.\texttt{Purchase}.\overline{\texttt{Accepted}}$ so as to avoid, say, $\texttt{Login}.\overline{\texttt{InvalidLogin}}$.

Application to service discovery

Search for services compliant with a client ρ:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid \rho \dashv f(\sigma)\}$$

Call one of the services in the result by using the associated filter.

More efficient using subcontracts and caching

1. Compute $\rho^{\perp}$ the "canonical server" of ρ
2. Return all compatible servers:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma\} \qquad \text{(cached)}$$

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

$$\texttt{discover}(\rho, g) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma \ \wedge g \leq f\}$$

E.g.: provide at least $\texttt{Login}.\overline{\texttt{ValidLogin}}.\texttt{Query}.\overline{\texttt{Catalog}}.\texttt{Purchase}.\overline{\texttt{Accepted}}$ so as to avoid, say, $\texttt{Login}.\overline{\texttt{InvalidLogin}}$.

Application to service discovery

Search for services compliant with a client ρ:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid \rho \dashv f(\sigma)\}$$

Call one of the services in the result by using the associated filter.

More efficient using subcontracts and caching

1. Compute $\rho^{\perp}$ the "canonical server" of ρ
2. Return all compatible servers:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma\} \qquad \text{(cached)}$$

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

$$\texttt{discover}(\rho, g) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma \;\wedge g \leq f\}$$

E.g.: provide at least $\texttt{Login}.\overline{\texttt{ValidLogin}}.\texttt{Query}.\overline{\texttt{Catalog}}.\texttt{Purchase}.\overline{\texttt{Accepted}}$ so as to avoid, say, $\texttt{Login}.\overline{\texttt{InvalidLogin}}$.

Application to service discovery

Search for services compliant with a client ρ:

$$\mathtt{discover}(\rho) = \{(\sigma, f) \mid \rho \dashv f(\sigma)\}$$

Call one of the services in the result by using the associated filter.

More efficient using subcontracts and caching

1. Compute $\rho^{\perp}$ the "canonical server" of ρ
2. Return all compatible servers:

$$\mathtt{discover}(\rho) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma\} \qquad \text{(cached)}$$

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

$$\mathtt{discover}(\rho, g) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma \ \wedge g \leq f\}$$

E.g.: provide at least `Login.ValidLogin.Query.Catalog.Purchase.Accepted` so as to avoid, say, `Login.InvalidLogin`.

Application to service discovery

Search for services compliant with a client ρ:

$$\mathtt{discover}(\rho) = \{(\sigma, f) \mid \rho \dashv f(\sigma)\}$$

Call one of the services in the result by using the associated filter.

More efficient using subcontracts and caching

1. Compute $\rho^{\perp}$ the "canonical server" of ρ
2. Return all compatible servers:

$$\mathtt{discover}(\rho) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma\} \qquad \text{(cached)}$$

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

$$\mathtt{discover}(\rho, g) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma \ \wedge g \leq f\}$$

E.g.: provide at least $\mathtt{Login}.\overline{\mathtt{ValidLogin}}.\mathtt{Query}.\overline{\mathtt{Catalog}}.\mathtt{Purchase}.\overline{\mathtt{Accepted}}$ so as to avoid, say, $\mathtt{Login}.\overline{\mathtt{InvalidLogin}}$.

Application to service discovery

Search for services compliant with a client ρ:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid \rho \dashv f(\sigma)\}$$

Call one of the services in the result by using the associated filter.

More efficient using subcontracts and caching

1. Compute $\rho^\perp$ the "canonical server" of ρ
2. Return all compatible servers:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid f : \rho^\perp \preceq \sigma\} \qquad \text{(cached)}$$

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

$$\texttt{discover}(\rho, g) = \{(\sigma, f) \mid f : \rho^\perp \preceq \sigma \ \wedge g \leq f\}$$

E.g.: provide at least $\texttt{Login}.\overline{\texttt{ValidLogin}}.\texttt{Query}.\overline{\texttt{Catalog}}.\texttt{Purchase}.\overline{\texttt{Accepted}}$ so as to avoid, say, $\texttt{Login}.\overline{\texttt{InvalidLogin}}$.

Application to service discovery

Search for services compliant with a client ρ:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid \rho \dashv f(\sigma)\}$$

Call one of the services in the result by using the associated filter.

More efficient using subcontracts and caching

1. Compute $\rho^{\perp}$ the "canonical server" of ρ
2. Return all compatible servers:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma\} \qquad \text{(cached)}$$

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

$$\texttt{discover}(\rho, g) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma \ \wedge g \leq f\}$$

E.g.: provide at least $\texttt{Login}.\overline{\texttt{ValidLogin}}.\texttt{Query}.\overline{\texttt{Catalog}}.\texttt{Purchase}.\overline{\texttt{Accepted}}$ so as to avoid, say, $\texttt{Login}.\overline{\texttt{InvalidLogin}}$.

Application to service discovery

Search for services compliant with a client ρ:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid \rho \dashv f(\sigma)\}$$

Call one of the services in the result by using the associated filter.

More efficient using subcontracts and caching

1. Compute $\rho^{\perp}$ the "canonical server" of ρ
2. Return all compatible servers:

$$\texttt{discover}(\rho) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma\} \qquad \text{(cached)}$$

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

$$\texttt{discover}(\rho, g) = \{(\sigma, f) \mid f : \rho^{\perp} \preceq \sigma \ \wedge g \leq f\}$$

E.g.: provide at least $\texttt{Login}.\overline{\texttt{ValidLogin}}.\texttt{Query}.\overline{\texttt{Catalog}}.\texttt{Purchase}.\overline{\texttt{Accepted}}$ so as to avoid, say, $\texttt{Login}.\overline{\texttt{InvalidLogin}}$.

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
    - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
    - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
    - Recursion (almost done)
    - Application to WS-BPEL (almost done: pretty easy)
    - Integration with session types (in progress)
    - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Full details in the forthcoming POPL 2008 paper

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Full details in the forthcoming POPL 2008 paper

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Full details in the forthcoming POPL 2008 paper

Conclusion

- The approach reconciles two desiderata:
  - upgrade services by more operations (width subtyping), longer interaction patterns (depth subtyping) and/or more deterministic ones;
  - upgrade transparency for old clients
- Technical device: filters
- Filters as explicit coercions between contracts and as proofs for subcontracting.
- Algorithmic counterpart obtained via cut-elimination.
- Theory independent from the language(s) used to implement clients and services.
- Applications: specification and discovery of services.
- Future work
  - Recursion (almost done)
  - Application to WS-BPEL (almost done: pretty easy)
  - Integration with session types (in progress)
  - Choreography, Messages, and higher order channels

Full details in the forthcoming POPL 2008 paper