

# A Theory of Contracts for Web Services

Giuseppe Castagna, Nils Gesbert, Luca Padovani

Université Paris 7, Université Paris Sud, Università di Urbino

FLACOS '07

# 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...)

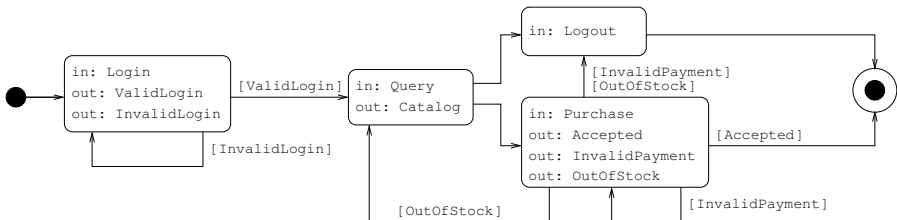
# Summary

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

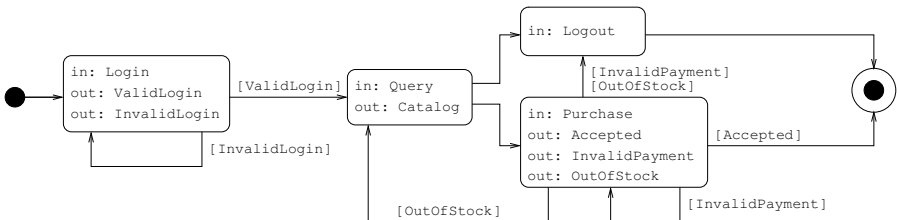


# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions  
 $\text{Query.Catalog}$
- 2  
 $\text{Login}(\overline{\text{ValidLogin}} \dots \oplus \overline{\text{InvalidLogin}} \dots)$
- 3  
 $\text{Query.Catalog}(\overline{\text{Logout}} \dots + \overline{\text{Purchase}} \dots)$

# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions

`Query.Catalog`

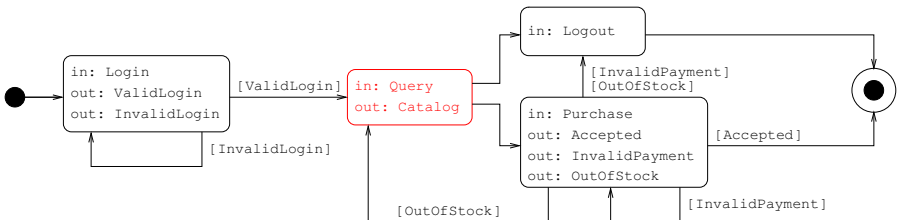


`Login.(ValidLogin... ⊕ InvalidLogin...)`



`Query.Catalog.(Logout... + Purchase...)`

# What is a contract?



## 1 Describes sequences of INPUT/OUTPUT actions

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

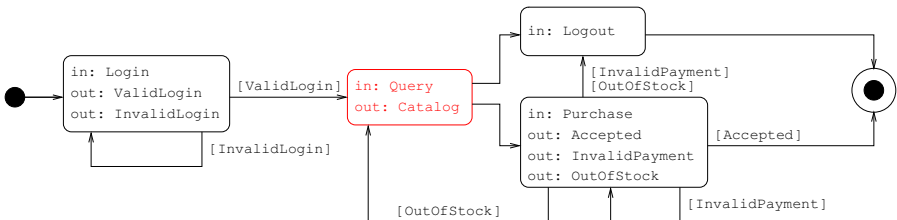
2

$\overline{\text{Login.}(\overline{\text{ValidLogin}} \dots \oplus \overline{\text{InvalidLogin}} \dots)}$

3

$\overline{\text{Query.Catalog.}(\overline{\text{Logout}} \dots + \overline{\text{Purchase}} \dots)}$

# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions

Query.Catalog

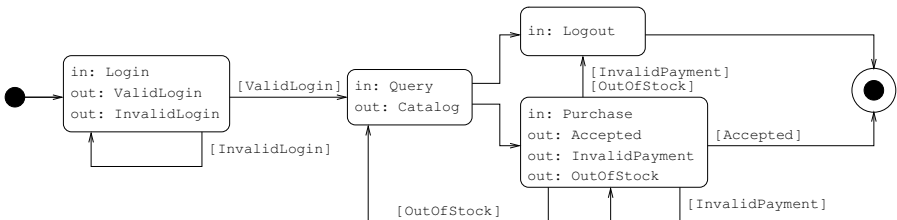
2

Login.(ValidLogin... ⊕ InvalidLogin...)

3

Query.Catalog.(Logout... + Purchase...)

# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions

Query.Catalog

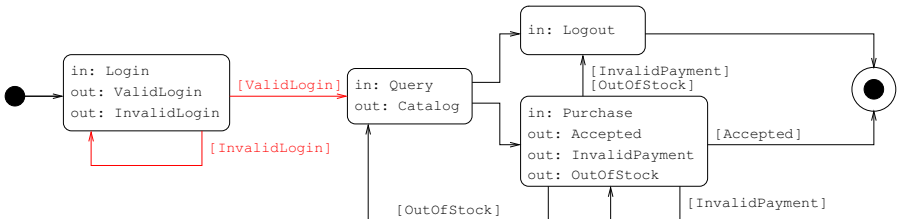
- 2 Describes possible internal choices

Login.(ValidLogin...  $\oplus$  InvalidLogin...)

- 3

Query.Catalog.(Logout... + Purchase...)

# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions

Query.Catalog

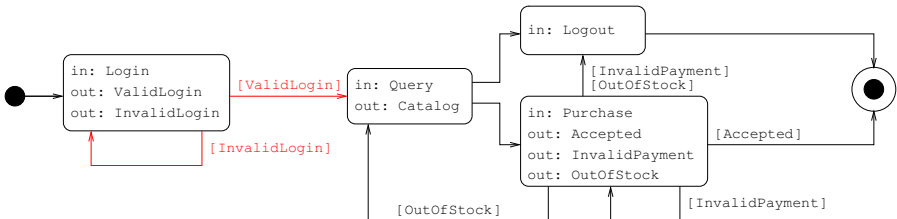
- 2 Describes possible internal choices

Login.(ValidLogin...  $\oplus$  InvalidLogin...)

- 3

Query.Catalog.(Logout... + Purchase...)

# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions

Query.Catalog

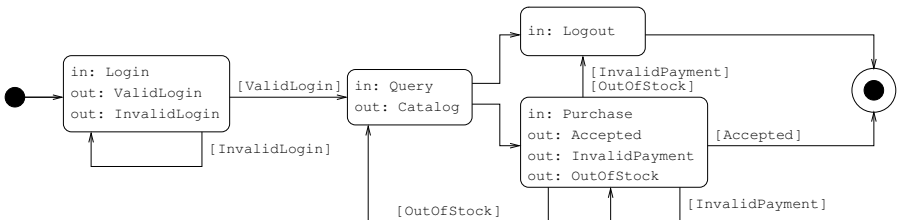
- 2 Describes possible internal choices

Login.(ValidLogin. ...  $\oplus$  InvalidLogin. ...)

- 3

Query.Catalog.(Logout. ... + Purchase. ...)

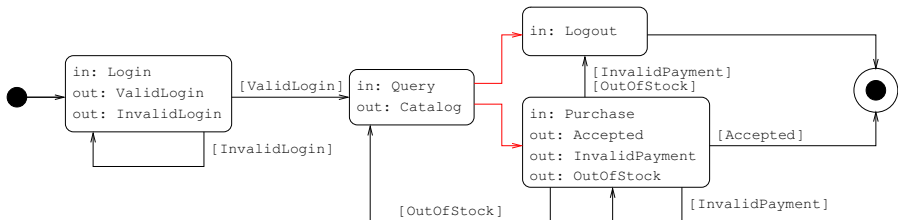
# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions  
 $\text{Query.Catalog}$
- 2 Describes possible internal choices  
 $\text{Login.}(\overline{\text{ValidLogin}} \dots \oplus \overline{\text{InvalidLogin}} \dots)$
- 3 Describes available external choices  
 $\text{Query.Catalog.}(\overline{\text{Logout}} \dots + \overline{\text{Purchase}} \dots)$

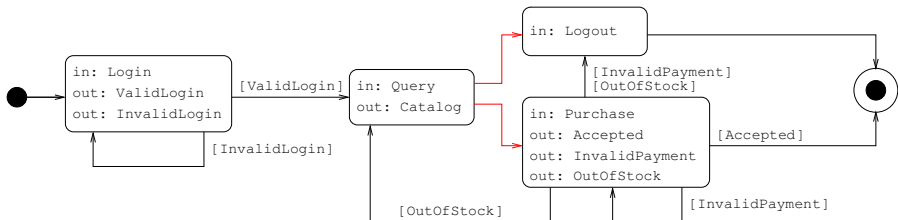


# What is a contract?



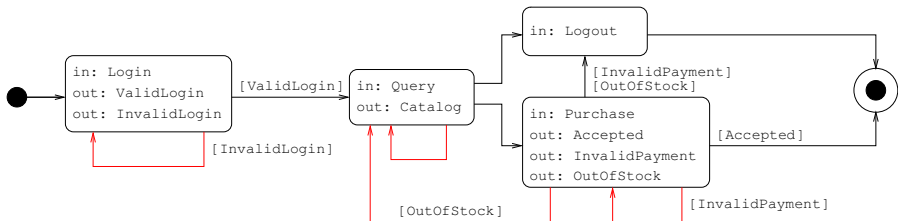
- 1 Describes sequences of INPUT/OUTPUT actions  
Query.Catalog
- 2 Describes possible internal choices  
Login.(ValidLogin...  $\oplus$  InvalidLogin...)
- 3 Describes available external choices  
Query.Catalog.(Logout... + Purchase...)

# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions  
Query.Catalog
- 2 Describes possible internal choices  
Login.(ValidLogin. ...  $\oplus$  InvalidLogin. ...)
- 3 Describes available external choices  
Query.Catalog.(Logout. ... + Purchase. ...)

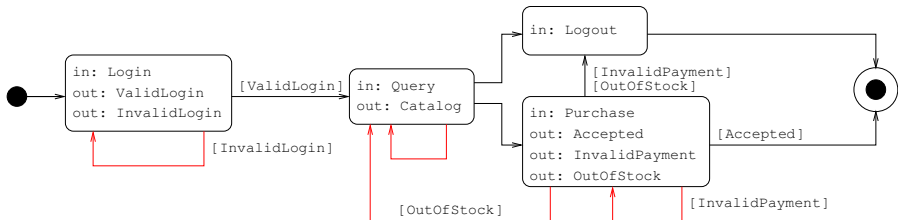
# What is a contract?



- 1 Describes sequences of INPUT/OUTPUT actions  
 $\text{Query.Catalog}$
- 2 Describes possible internal choices  
 $\text{Login.}(\overline{\text{ValidLogin}} \dots \oplus \overline{\text{InvalidLogin}} \dots)$
- 3 Describes available external choices  
 $\text{Query.Catalog.}(\overline{\text{Logout}} \dots + \overline{\text{Purchase}} \dots)$

Note that the contract is recursive

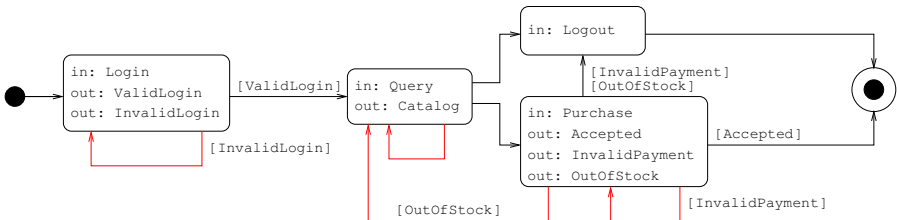
# What is a contract?



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

Note that the contract is recursive

# What is a contract?



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

We do not consider recursion in this work

# A formal contract language

**contracts**     $\sigma ::=$

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

**actions**     $\alpha ::=$

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

Names represent  
**types, operations, ...**

# A formal contract language

**contracts**  $\sigma ::=$

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

**actions**  $\alpha ::=$

$a$  (receive)  
 $\bar{a}$  (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**  $\sigma ::=$

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

**actions**  $\alpha ::=$

$a$  (receive)  
 $\bar{a}$  (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**  $\sigma ::=$

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

**actions**  $\alpha ::=$

$a$  (receive)  
 $\bar{a}$  (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

$$\alpha.\sigma \xrightarrow{\alpha} \sigma$$

$\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2$	$\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}$
$\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2$	$\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1$
$\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2$	$\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}$
$\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2$	$\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1$

- 2 HOW actions are offered

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

# Semantics

The contract of a process describes

- 1 WHICH actions the process offers

$$\alpha.\sigma \xrightarrow{\alpha} \sigma$$

$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2}{\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2}$	$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}}{\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1}$
$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2}{\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2}$	$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}}{\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1}$

- 2 HOW actions are offered

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

# Semantics

The contract of a process describes

- ① WHICH actions the process offers

$$\alpha.\sigma \xrightarrow{\alpha} \sigma$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2}{\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2}$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2}{\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2}$$

$(\sigma \xrightarrow{\alpha}: \text{“}\sigma \text{ can emit } \alpha\text{”})$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}}{\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1}$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}}{\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1}$$

- ② HOW actions are offered

$$\emptyset \Downarrow \emptyset$$

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

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

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

# Semantics

The contract of a process describes

- ① WHICH actions the process offers

$$\alpha.\sigma \xrightarrow{\alpha} \sigma$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2}{\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2}$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2}{\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2}$$

$(\sigma \xrightarrow{\alpha}: \text{“}\sigma \text{ can emit } \alpha\text{”})$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}}{\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1}$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}}{\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1}$$

- ② HOW actions are offered

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

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

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

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

# Semantics

The contract of a process describes

- ① WHICH actions the process offers

$$\alpha.\sigma \xrightarrow{\alpha} \sigma$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2}{\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2}$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \xrightarrow{\alpha} \sigma'_2}{\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1 \oplus \sigma'_2}$$

$(\sigma \xrightarrow{\alpha}: \text{“}\sigma \text{ can emit } \alpha\text{”})$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}}{\sigma_1 + \sigma_2 \xrightarrow{\alpha} \sigma'_1}$$

$$\frac{\sigma_1 \xrightarrow{\alpha} \sigma'_1 \quad \sigma_2 \not\xrightarrow{\alpha}}{\sigma_1 \oplus \sigma_2 \xrightarrow{\alpha} \sigma'_1}$$

- ② HOW actions are offered

$(\sigma \Downarrow R: \text{“}\sigma \text{ can offer to choose in } R\text{”})$

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

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

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

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

# Semantics

For instance

$$a \oplus b \xrightarrow{a} \mathbf{0}$$

$$a \oplus b \xrightarrow{b} \mathbf{0}$$

$$a + b \xrightarrow{a} \mathbf{0}$$

$$a + b \xrightarrow{b} \mathbf{0}$$

but

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

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

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

## Relating clients and services: **compliance**

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

- $a.\wp + b.\wp \dashv \bar{a} \oplus \bar{b}$
- $a.\wp + b.\wp \dashv \bar{a}$
- $a.\wp \oplus b.\wp \dashv \bar{a}.c + \bar{b}.d$
- $a.\wp \oplus b.\wp \dashv \bar{a} \oplus \bar{b}$

( $\wp$  indicates client's satisfaction).



## Relating clients and services: **compliance**

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

- $a.\text{✌} + b.\text{✌} \dashv \bar{a} \oplus \bar{b}$
- $a.\text{✌} + b.\text{✌} \dashv \bar{a}$
- $a.\text{✌} \oplus b.\text{✌} \dashv \bar{a}.c + \bar{b}.d$
- $a.\text{✌} \oplus b.\text{✌} \dashv \bar{a} \oplus \bar{b}$

(✌ indicates client's satisfaction).

# Relating clients and services: **compliance**

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

- $a.\text{✌} + b.\text{✌} \dashv \bar{a} \oplus \bar{b}$
- $a.\text{✌} + b.\text{✌} \dashv \bar{a}$
- $a.\text{✌} \oplus b.\text{✌} \dashv \bar{a}.c + \bar{b}.d$
- $a.\text{✌} \oplus b.\text{✌} \not\vdash \bar{a} \oplus \bar{b}$

(✌ indicates client's satisfaction).

Formally  $\rho \dashv \sigma$  iff

- ①  $\rho \Downarrow R$ ,  $\sigma \Downarrow S$ , and  $\bar{R} \cap S = \emptyset$  imply  $\text{✌} \in R$
- ②  $\rho \xrightarrow{\bar{\alpha}} \rho'$  and  $\sigma \xrightarrow{\alpha} \sigma'$  imply  $\rho' \dashv \sigma'$

# Relating different services: **subcontract**

## Intuition

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

- 1 Is more deterministic:

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

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

- 2 Offers more choices:

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

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

- 3 Offers longer interaction patterns:

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

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

# Relating different services: **subcontract**

## Intuition

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

### 1 Is more deterministic:

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

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

### 2 Offers more choices:

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

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

### 3 Offers longer interaction patterns:

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

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

# Relating different services: **subcontract**

## Intuition

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

### 1 Is more deterministic:

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

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

### 2 Offers more choices:

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

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

### 3 Offers longer interaction patterns:

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

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

# Relating different services: **subcontract**

## Intuition

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

- ① **Is more deterministic:**

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

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

- ② **Offers more choices:**

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

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

- ③ **Offers longer interaction patterns:**

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

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

# Relating different services: **subcontract**

## Intuition

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

- ① **Is more deterministic:**

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

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

- ② **Offers more choices:**

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

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

- ③ **Offers longer interaction patterns:**

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

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

# Relating different services: **subcontract**

## Intuition

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

- ① **Is more deterministic:**

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

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

- ② **Offers more choices:**

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

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

- ③ **Offers longer interaction patterns:**

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

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



# Relating different services: **subcontract**

## Intuition

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

- ① **Is more deterministic:**

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

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

- ② **Offers more choices:**

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

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

- ③ **Offers longer interaction patterns:**

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

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

# Relating different services: **subcontract**

## Intuition

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

- ① **Is more deterministic:**

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

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

- ② **Offers more choices:**

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

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

- ③ **Offers longer interaction patterns:**

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

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

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

# Compliance and Subcontract mismatch

- Note:

$$\bar{a} \oplus \bar{b}.c \stackrel{\textcircled{1}}{\preceq} \bar{a} \stackrel{\textcircled{2}}{\preceq} \bar{a} + \bar{b}.d$$

- but for a client  $a.\bar{b} + b.\bar{c}.\bar{d}$ :

$$a.\bar{b} + b.\bar{c}.\bar{d} \not\vdash \bar{a} \oplus \bar{b}.c \quad a.\bar{b} + b.\bar{c}.\bar{d} \not\vdash \bar{a} + \bar{b}.d$$

- Can we replace a server  $\bar{a} \oplus \bar{b}.c$  for a  $\bar{a} + \bar{b}.d$  one (i.e.  $\bar{a} \oplus \bar{b}.c :> \bar{a} + \bar{b}.d$ )?
- YES if “:>” uses explicit coercions (rather than implicit ones)

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

# Compliance and Subcontract mismatch

- Note:

$$\bar{a} \oplus \bar{b}.c \stackrel{\textcircled{1}}{\preceq} \bar{a} \stackrel{\textcircled{2}}{\preceq} \bar{a} + \bar{b}.d$$

- but for a client  $a.\heartsuit + b.\bar{c}.\heartsuit$ :

$$a.\heartsuit + b.\bar{c}.\heartsuit \not\vdash \bar{a} \oplus \bar{b}.c \quad a.\heartsuit + b.\bar{c}.\heartsuit \not\vdash \bar{a} + \bar{b}.d$$

- Can we replace a server  $\bar{a} \oplus \bar{b}.c$  for a  $\bar{a} + \bar{b}.d$  one (i.e.  $\bar{a} \oplus \bar{b}.c :> \bar{a} + \bar{b}.d$ )?
- YES if “:>” uses explicit coercions (rather than implicit ones)

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

# Compliance and Subcontract mismatch

- Note:

$$\bar{a} \oplus \bar{b}.c \stackrel{\textcircled{1}}{\preceq} \bar{a} \stackrel{\textcircled{2}}{\preceq} \bar{a} + \bar{b}.d$$

- but for a client  $a.\heartsuit + b.\bar{c}.\heartsuit$ :

$$a.\heartsuit + b.\bar{c}.\heartsuit \not\vdash \bar{a} \oplus \bar{b}.c \quad a.\heartsuit + b.\bar{c}.\heartsuit \not\vdash \bar{a} + \bar{b}.d$$

- Can we replace a server  $\bar{a} \oplus \bar{b}.c$  for a  $\bar{a} + \bar{b}.d$  one (i.e.  $\bar{a} \oplus \bar{b}.c :> \bar{a} + \bar{b}.d$ )?
- YES if “:>” uses explicit coercions (rather than implicit ones)

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

# Compliance and Subcontract mismatch

- Note:

$$\bar{a} \oplus \bar{b}.c \stackrel{\textcircled{1}}{\preceq} \bar{a} \stackrel{\textcircled{2}}{\preceq} \bar{a} + \bar{b}.d$$

- but for a client  $a.\heartsuit + b.\bar{c}.\heartsuit$ :

$$a.\heartsuit + b.\bar{c}.\heartsuit \not\vdash \bar{a} \oplus \bar{b}.c \quad a.\heartsuit + b.\bar{c}.\heartsuit \not\vdash \bar{a} + \bar{b}.d$$

- Can we replace a server  $\bar{a} \oplus \bar{b}.c$  for a  $\bar{a} + \bar{b}.d$  one (i.e.  $\bar{a} \oplus \bar{b}.c :> \bar{a} + \bar{b}.d$ )?
- YES if “:>” uses explicit coercions (rather than implicit ones)

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

# Compliance and Subcontract mismatch

- Note:

$$\bar{a} \oplus \bar{b}.c \stackrel{\textcircled{1}}{\preceq} \bar{a} \stackrel{\textcircled{2}}{\preceq} \bar{a} + \bar{b}.d$$

- but for a client  $a.\heartsuit + b.\bar{c}.\heartsuit$ :

$$a.\heartsuit + b.\bar{c}.\heartsuit \not\vdash \bar{a} \oplus \bar{b}.c \quad a.\heartsuit + b.\bar{c}.\heartsuit \not\vdash \bar{a} + \bar{b}.d$$

- Can we replace a server  $\bar{a} \oplus \bar{b}.c$  for a  $\bar{a} + \bar{b}.d$  one (i.e.  $\bar{a} \oplus \bar{b}.c :> \bar{a} + \bar{b}.d$ )?
- YES if “:>” uses explicit coercions (rather than implicit ones)

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

# Gluing compliance and subcontracting: **Filters**

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

Transition relation of filters

$$\coprod_{\alpha \in A} \alpha.f_{\alpha} \xrightarrow{\beta} f_{\beta} \quad \text{if } \beta \in A$$

Contract coercion through a filter

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

Property

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



# Gluing compliance and subcontracting: **Filters**

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

Transition relation of filters

$$\coprod_{\alpha \in A} \alpha.f_{\alpha} \xrightarrow{\beta} f_{\beta} \quad \text{if } \beta \in A$$

Contract coercion through a filter

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

Property

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

# Gluing compliance and subcontracting: **Filters**

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

Transition relation of filters

$$\coprod_{\alpha \in A} \alpha.f_{\alpha} \xrightarrow{\beta} f_{\beta} \quad \text{if } \beta \in A$$

Contract coercion through a filter

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

Property

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

# Gluing compliance and subcontracting: **Filters**

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

Transition relation of filters

$$\coprod_{\alpha \in A} \alpha.f_{\alpha} \xrightarrow{\beta} f_{\beta} \quad \text{if } \beta \in A$$

Contract coercion through a filter

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

Property

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

# 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*

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

Corollary

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

# 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*

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

Corollary

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

# 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*

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

Corollary

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

# 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*

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

Corollary

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

# 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*

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

Corollary

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



# 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*

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

Corollary

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

# 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*

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

## Corollary

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

# Application to languages

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

## 1 An LTS

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

$\mu$  is either a visible action or an invisible  $\tau$  action

## 2 A type system

$$\vdash P : \sigma$$

$\sigma$  is a contract

## 3 The latter abstracts the former:

- 1 If  $\vdash P : \sigma$  and  $\sigma \xrightarrow{\alpha}$ , then  $P \xrightarrow{\alpha}$
- 2 If  $\vdash P : \sigma$  and  $P \xrightarrow{\mu} P'$  then  $\vdash P' : \sigma'$  and
  - if  $\mu = \tau$ , then  $\sigma \sqsubseteq_{\text{must}} \sigma'$
  - if  $\mu = \alpha$ , then  $\sigma \xrightarrow{\alpha} \sqsubseteq_{\text{must}} \sigma'$

$\sqsubseteq_{\text{must}}$  measures non-determinism

# Application to languages

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

## 1 An LTS

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

$\mu$  is either a visible action or an invisible  $\tau$  action

## 2 A type system

$$\vdash P : \sigma$$

$\sigma$  is a contract

## 3 The latter abstracts the former:

- 1 If  $\vdash P : \sigma$  and  $\sigma \xrightarrow{\alpha}$ , then  $P \xrightarrow{\alpha}$
- 2 If  $\vdash P : \sigma$  and  $P \xrightarrow{\mu} P'$  then  $\vdash P' : \sigma'$  and
  - if  $\mu = \tau$ , then  $\sigma \sqsubseteq_{\text{must}} \sigma'$
  - if  $\mu = \alpha$ , then  $\sigma \xrightarrow{\alpha} \sqsubseteq_{\text{must}} \sigma'$

$\sqsubseteq_{\text{must}}$  measures non-determinism

# Application to languages

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

## ① An LTS

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

$\mu$  is either a visible action or an invisible  $\tau$  action

## ② A type system

$$\vdash P : \sigma$$

$\sigma$  is a contract

## ③ The latter abstracts the former:

- ① If  $\vdash P : \sigma$  and  $\sigma \xrightarrow{\alpha}$ , then  $P \xrightarrow{\alpha}$
- ② If  $\vdash P : \sigma$  and  $P \xrightarrow{\mu} P'$  then  $\vdash P' : \sigma'$  and
  - if  $\mu = \tau$ , then  $\sigma \sqsubseteq_{\text{must}} \sigma'$
  - if  $\mu = \alpha$ , then  $\sigma \xrightarrow{\alpha} \sqsubseteq_{\text{must}} \sigma'$

$\sqsubseteq_{\text{must}}$  measures non-determinism

# Application to languages

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

## ① An LTS

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

$\mu$  is either a visible action or an invisible  $\tau$  action

## ② A type system

$$\vdash P : \sigma$$

$\sigma$  is a contract

## ③ The latter abstracts the former:

- ① If  $\vdash P : \sigma$  and  $\sigma \xrightarrow{\alpha}$ , 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  $\sigma \xrightarrow{\alpha} \sqsubseteq_{must} \sigma'$

$\sqsubseteq_{must}$  measures non-determinism

# Process compliance

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

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

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

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  $Q \xrightarrow{\bar{\alpha}} Q'$  then  $P \parallel Q \longrightarrow P' \parallel Q'$  (plus  $\tau$ -moves)
- Client  $P$  complies with server  $Q$  (noted  $P \dashv Q$ ) if
  - if  $P \xrightarrow{\mu}$ , then  $\mu = \bar{\nu}$  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

$$\begin{array}{c}
 \text{(FILTER1)} \\
 \frac{P \xrightarrow{\alpha} P' \quad f \vdash \xrightarrow{\alpha} f'}{f[P] \xrightarrow{\alpha} f'[P']}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(FILTER2)} \\
 \frac{P \xrightarrow{\tau} P'}{f[P] \xrightarrow{\tau} f[P']}
 \end{array}$$

Typing rules for filters

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

“Subject reduction” still holds

# Process Filtering

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

Transition rules for filters

$$\begin{array}{c} \text{(FILTER1)} \\ \frac{P \xrightarrow{\alpha} P' \quad f \vdash \alpha \rightarrow f'}{f[P] \xrightarrow{\alpha} f'[P']} \end{array} \qquad \begin{array}{c} \text{(FILTER2)} \\ \frac{P \xrightarrow{\tau} P'}{f[P] \xrightarrow{\tau} f[P']} \end{array}$$

Typing rules for filters

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

“Subject reduction” still holds

*If  $\vdash P : \rho$  and  $\vdash Q : \sigma$  and  $f : \rho \prec \sigma$ , then  $P \rightarrow f[Q]$*

# Process Filtering

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

Transition rules for filters

$$\begin{array}{c} \text{(FILTER1)} \\ \frac{P \xrightarrow{\alpha} P' \quad f \vdash^{\alpha} f'}{f[P] \xrightarrow{\alpha} f'[P']} \end{array} \qquad \begin{array}{c} \text{(FILTER2)} \\ \frac{P \xrightarrow{\tau} P'}{f[P] \xrightarrow{\tau} f[P']} \end{array}$$

Typing rules for filters

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

“Subject reduction” still holds

Theorem (Process filtering)

*If  $\vdash P : \rho$  and  $\vdash Q : \sigma$  and  $f : \rho \preceq \sigma$ , then  $P \dashv f[Q]$*

# Process Filtering

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

Transition rules for filters

$$\begin{array}{c} \text{(FILTER1)} \\ \frac{P \xrightarrow{\alpha} P' \quad f \vdash^{\alpha} f'}{f[P] \xrightarrow{\alpha} f'[P']} \end{array} \qquad \begin{array}{c} \text{(FILTER2)} \\ \frac{P \xrightarrow{\tau} P'}{f[P] \xrightarrow{\tau} f[P']} \end{array}$$

Typing rules for filters

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

“Subject reduction” still holds

Theorem (Process filtering)

*If  $\vdash P : \rho$  and  $\vdash Q : \sigma$  and  $f : \rho \preceq \sigma$ , then  $P \dashv f[Q]$*

# Application to service discovery

Search for services compliant with a client  $\rho$ :

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

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  $\rho$
- ② Return all compatible servers:

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

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

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

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  $\rho$ :

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

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  $\rho$

② Return all compatible servers:

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

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

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

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  $\rho$ :

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

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  $\rho$

② Return all compatible servers:

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

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

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

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  $\rho$ :

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

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  $\rho$

② Return all compatible servers:

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

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

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

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  $\rho$ :

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

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  $\rho$
- ② Return all compatible servers:

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

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

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

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  $\rho$ :

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

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  $\rho$
- ② Return all compatible servers:

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

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

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

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  $\rho$ :

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

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  $\rho$
- ② Return all compatible servers:

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

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

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

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  $\rho$ :

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

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  $\rho$
- ② Return all compatible servers:

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

Finer-grained searches

Define some minimal behaviour that must not be filtered out:

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

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

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography



# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

# 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
    - Messages and higher order channels
    - Integration with session types
    - Choreography

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography



# 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
  - Messages and higher order channels
  - Integration with session types
  - Choreography

Full details in a forthcoming POPL 2008 paper