Contract-based Discovery and Adaptation of Web Services

Luca Padovani

Jointly with
Samuele Carpineti, Giuseppe Castagna, Nils Gesbert,
Cosimo Laneve

9th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Web Services

Contracts for Web services

We've got...

- behavioral descriptions of Web services (wsdl, wscl, ws-bpel, put your favorite technology here)
- repositories of Web services descriptions (uddi)

We'd like to...

- look for Web services with a given behavior
- see if it's safe to replace a service with another one
- if not, see whether we can *adapt* one service to safely replace another

We're gonna use...

- contracts = abstractions of Web services' behavior

Contracts for Web services

We've got. . .

- behavioral descriptions of Web services (wsdl, wscl, ws-bpel, put your favorite technology here)
- repositories of Web services descriptions (uddi)

We'd like to. . .

- look for Web services with a given behavior
- see if it's safe to replace a service with another one
- if not, see whether we can *adapt* one service to safely replace another

We're gonna use. . .

- contracts = abstractions of Web services' behavior

Contracts for Web services

We've got. . .

- behavioral descriptions of Web services (wsdl, wscl, ws-bpel, put your favorite technology here)
- repositories of Web services descriptions (uddi)

We'd like to. . .

- look for Web services with a given behavior
- see if it's safe to replace a service with another one
- if not, see whether we can *adapt* one service to safely replace another

We're gonna use. . .

- contracts = abstractions of Web services' behavior

Finding Web services by contract

Compliance = client's satisfaction

$$\rho \dashv \sigma$$

Running a query with *compliance*

$$\mathcal{Q}(\rho) = \{\sigma \mid \rho \dashv \sigma\}$$

Running a query with *duality* $\rho^\perp$ and *subcontract* $\sigma \preceq \tau$

$$\mathcal{Q}(\rho) = \{\sigma \mid \rho^\perp \preceq \sigma\}$$

Finding Web services by contract

Compliance = client's satisfaction

$$\rho \dashv \sigma$$

Running a query with *compliance*

$$\mathcal{Q}(\rho) = \{\sigma \mid \rho \dashv \sigma\}$$

Running a query with *duality* $\rho^{\perp}$ and *subcontract* $\sigma \preceq \tau$

$$\mathcal{Q}(\rho) = \{\sigma \mid \rho^{\perp} \preceq \sigma\}$$

The quest for $\preceq$

Desired properties of $\preceq$

- reduction of nondeterminism ($a \oplus b \preceq a$)
- extension of functionalities ($a \preceq a + b$)
- some permutation of messages ($a.c \preceq c.a$)

The problem

- *reduction* alone is too strict
- *extension* is unsafe
- *extension;reduction* is not transitive
- *permutation* is not allowed

The solution

- use (simple) orchestrators

The quest for $\preceq$

Desired properties of $\preceq$

- reduction of nondeterminism ($a \oplus b \preceq a$)
- extension of functionalities ($a \preceq a + b$)
- some permutation of messages ($a.c \preceq c.a$)

The problem

- *reduction* alone is too strict
- *extension* is unsafe
- *extension;reduction* is not transitive
- *permutation* is not allowed

The solution

- use (simple) orchestrators

The quest for $\preceq$

Desired properties of $\preceq$

- reduction of nondeterminism ($a \oplus b \preceq a$)
- extension of functionalities ($a \preceq a + b$)
- some permutation of messages ($a.c \preceq c.a$)

The problem

- *reduction* alone is too strict
- *extension* is unsafe
- *extension;reduction* is not transitive
- *permutation* is not allowed

The solution

- use (simple) orchestrators

Summary

1. contracts
2. simple orchestrators
3. simple orchestrators with buffers
4. duality
5. recursive behaviors
6. orchestrator synthesis
7. related and ongoing work

Oh no! More ws-bpel!

```
<process>
  <sequence>
    <receive operation="Order" variable="Request"/>

    <flow>
      <invoke operation="Deposit" inputVariable="Request" outputVariable="Deposit"/>
      <invoke operation="Charge" inputVariable="Request" outputVariable="Charge"/>
    </flow>

    <switch>
      <case condition="getVariableData(Deposit) == true && getVariableData(Charge) == true)">
        <invoke operation="Ship" inputVariable="Request"/>
        <reply operation="Order" value="OK"/>
      </case>

      <case condition="getVariableData(Charge) == true)">
        <invoke operation="Refund" inputVariable="Request"/>
        <reply operation="Order" value="NO"/>
      </case>

      <otherwise>
        <reply operation="Order" value="NO"/>
      </otherwise>
    </switch>
  </sequence>
</process>
```

Oh no! More ws-bpel!

```
<process>
  <sequence>
    <receive operation="Order" variable="Request"/>

    <flow>
      <invoke operation="Deposit" inputVariable="Request" outputVariable="Deposit"/>
      <invoke operation="Charge" inputVariable="Request" outputVariable="Charge"/>
    </flow>

    <switch>
      <case condition="getVariableData(Deposit) == true && getVariableData(Charge) == true)">
        <invoke operation="Ship" inputVariable="Request"/>
        <reply operation="Order" value="OK"/>
      </case>

      <case condition="getVariableData(Charge) == true)">
        <invoke operation="Refund" inputVariable="Request"/>
        <reply operation="Order" value="NO"/>
      </case>

      <otherwise>
        <reply operation="Order" value="NO"/>
      </otherwise>
    </switch>
  </sequence>
</process>
```

Oh no! More ws-bpel!

```
<process>
  <sequence>
    <receive operation="Order" variable="Request"/>

    <flow>
      <invoke operation="Deposit" inputVariable="Request" outputVariable="Deposit"/>
      <invoke operation="Charge" inputVariable="Request" outputVariable="Charge"/>
    </flow>

    <switch>
      <case condition="getVariableData(Deposit) == true && getVariableData(Charge) == true)">
        <invoke operation="Ship" inputVariable="Request"/>
        <reply operation="Order" value="OK"/>
      </case>

      <case condition="getVariableData(Charge) == true)">
        <invoke operation="Refund" inputVariable="Request"/>
        <reply operation="Order" value="NO"/>
      </case>

      <otherwise>
        <reply operation="Order" value="NO"/>
      </otherwise>
    </switch>
  </sequence>
</process>
```

Oh no! More ws-bpel!

```
<process>
  <sequence>
    <receive operation="Order" variable="Request"/>

    <flow>
      <invoke operation="Deposit" inputVariable="Request" outputVariable="Deposit"/>
      <invoke operation="Charge" inputVariable="Request" outputVariable="Charge"/>
    </flow>

    <switch>
      <case condition="getVariableData(Deposit) == true && getVariableData(Charge) == true)">
        <invoke operation="Ship" inputVariable="Request"/>
        <reply operation="Order" value="OK"/>
      </case>

      <case condition="getVariableData(Charge) == true)">
        <invoke operation="Refund" inputVariable="Request"/>
        <reply operation="Order" value="NO"/>
      </case>

      <otherwise>
        <reply operation="Order" value="NO"/>
      </otherwise>
    </switch>
  </sequence>
</process>
```

Oh no! More ws-bpel!

```
<process>
  <sequence>
    <receive operation="Order" variable="Request"/>

    <flow>
      <invoke operation="Deposit" inputVariable="Request" outputVariable="Deposit"/>
      <invoke operation="Charge" inputVariable="Request" outputVariable="Charge"/>
    </flow>

    <switch>
      <case condition="getVariableData(Deposit) == true && getVariableData(Charge) == true)">
        <invoke operation="Ship" inputVariable="Request"/>
        <reply operation="Order" value="OK"/>
      </case>

      <case condition="getVariableData(Charge) == true)">
        <invoke operation="Refund" inputVariable="Request"/>
        <reply operation="Order" value="NO"/>
      </case>

      <otherwise>
        <reply operation="Order" value="NO"/>
      </otherwise>
    </switch>
  </sequence>
</process>
```

Oh no! More ws-bpel!

```
<process>
  <sequence>
    <receive operation="Order" variable="Request"/>

    <flow>
      <invoke operation="Deposit" inputVariable="Request" outputVariable="Deposit"/>
      <invoke operation="Charge" inputVariable="Request" outputVariable="Charge"/>
    </flow>

    <switch>
      <case condition="getVariableData(Deposit) == true && getVariableData(Charge) == true)">
        <invoke operation="Ship" inputVariable="Request"/>
        <reply operation="Order" value="OK"/>
      </case>

      <case condition="getVariableData(Charge) == true)">
        <invoke operation="Refund" inputVariable="Request"/>
        <reply operation="Order" value="NO"/>
      </case>

      <otherwise>
        <reply operation="Order" value="NO"/>
      </otherwise>
    </switch>
  </sequence>
</process>
```

What's in a contract

Actions

O	Order
D	Deposit
C	Charge
S	Ship
R	Refund

Traces

$$\{ODCSO, OCDSO, ODCRO, OCDRO, ODCO, OCDO\}$$

Branching points

- OD... and OC... is an *external choice*
- ...SO, ...RO, and ...O is an *internal choice*

What's in a contract

Actions

`O`	`Order`
`D`	`Deposit`
`C`	`Charge`
`S`	`Ship`
`R`	`Refund`

Traces

$$\{\texttt{ODCSO}, \texttt{OCDSO}, \texttt{ODCRO}, \texttt{OCDRO}, \texttt{ODCO}, \texttt{OCDO}\}$$

Branching points

- `OD`... and `OC`... is an *external choice*
- ...`SO`, ...`RO`, and ...`O` is an *internal choice*

What's in a contract

Actions

`O`	`Order`
`D`	`Deposit`
`C`	`Charge`
`S`	`Ship`
`R`	`Refund`

Traces

$$\{\texttt{ODCSO}, \texttt{OCDSO}, \texttt{ODCRO}, \texttt{OCDRO}, \texttt{ODCO}, \texttt{OCDO}\}$$

Branching points

- `OD`... and `OC`... is an *external choice*
- ...`SO`, ...`RO`, and ...`O` is an *internal choice*

Basic theory of contracts

Contracts

Syntax

σ	$::=$		**contract**	α	$::=$		**action**
		0	(null)			a	(input)
	$\mid$	$\alpha.\sigma$	(action prefix)		$\mid$	$\bar{a}$	(output)
	$\mid$	$\sigma + \sigma$	(external choice)				
	$\mid$	$\sigma \oplus \sigma$	(internal choice)				

Example

$$\mathtt{O}.(\overline{\mathtt{D}}.\overline{\mathtt{C}}.\mathtt{D}.\mathtt{C}.(\overline{\mathtt{S}}.\overline{\mathtt{O}} \oplus \overline{\mathtt{R}}.\overline{\mathtt{O}} \oplus \overline{\mathtt{O}}) + \overline{\mathtt{C}}.\overline{\mathtt{D}}.\mathtt{D}.\mathtt{C}.(\overline{\mathtt{S}}.\overline{\mathtt{O}} \oplus \overline{\mathtt{R}}.\overline{\mathtt{O}} \oplus \overline{\mathtt{O}}))$$

Operational semantics

$$\alpha.\sigma \xrightarrow{\alpha} \sigma \qquad \sigma \oplus \tau \longrightarrow \sigma \qquad \frac{\sigma \xrightarrow{\alpha} \sigma'}{\sigma + \tau \xrightarrow{\alpha} \sigma'} \qquad \frac{\sigma \longrightarrow \sigma'}{\sigma + \tau \longrightarrow \sigma' + \tau}$$

Compliance, formally

Systems

$$\rho \parallel \sigma$$

System transitions

$$\frac{\rho \longrightarrow \rho'}{\rho \parallel \sigma \longrightarrow \rho' \parallel \sigma} \qquad \frac{\sigma \longrightarrow \sigma'}{\rho \parallel \sigma \longrightarrow \rho \parallel \sigma'} \qquad \frac{\rho \xrightarrow{\overline{\alpha}} \rho' \qquad \sigma \xrightarrow{\alpha} \sigma'}{\rho \parallel \sigma \longrightarrow \rho' \parallel \sigma'}$$

Compliance

$$\rho \dashv \sigma \quad \stackrel{\text{def}}{\iff} \quad \rho \parallel \sigma \Longrightarrow \rho' \parallel \sigma' \not\longrightarrow \text{ implies } \rho' \xrightarrow{\mathsf{e}}$$

Compliance, examples

$$\bar{a}.\mathsf{e} \dashv ? \; a$$

$$\bar{a}.\mathsf{e} \oplus \bar{b}.\mathsf{e} \dashv a + b$$

$$\mathsf{e} \dashv \sigma$$

$$\bar{a}.\mathsf{e} \oplus \bar{b}.\mathsf{e} \dashv a \oplus b$$

$$\bar{a}.\mathsf{e} \dashv a \oplus 0$$

$$0 \dashv \sigma$$

Compliance, examples

$$\bar{a}.\mathsf{e} \dashv a \quad ☺$$

$$\bar{a}.\mathsf{e} \oplus \bar{b}.\mathsf{e} \dashv ? \; a + b$$

$$\mathsf{e} \dashv \sigma$$

$$\bar{a}.\mathsf{e} \oplus \bar{b}.\mathsf{e} \dashv a \oplus b$$

$$\bar{a}.\mathsf{e} \dashv a \oplus 0$$

$$0 \dashv \sigma$$

Compliance, examples

$$\overline{a}.\mathsf{e} \dashv a$$ ☺

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \dashv a + b$$ ☺

$$\mathsf{e} \dashv ? \ \sigma$$

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \dashv a \oplus b$$

$$\overline{a}.\mathsf{e} \dashv a \oplus 0$$

$$0 \dashv \sigma$$

Compliance, examples

$$\overline{a}.\mathsf{e} \dashv a \quad ☺$$

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \dashv a + b \quad ☺$$

$$\mathsf{e} \dashv \sigma \quad ☺$$

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \dashv ? \; a \oplus b$$

$$\overline{a}.\mathsf{e} \dashv a \oplus 0$$

$$0 \dashv \sigma$$

Compliance, examples

$\overline{a}.\mathsf{e}$	$\dashv$	a	☺
$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e}$	$\dashv$	$a + b$	☺
e	$\dashv$	σ	☺
$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e}$	$\dashv$?	$a \oplus b$	☹
$\overline{a}.\mathsf{e}$	$\dashv$?	$a \oplus 0$	
0	$\dashv$	σ	

Compliance, examples

$\overline{a}.\mathsf{e}$	$\dashv$	a	☺
$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e}$	$\dashv$	$a + b$	☺
e	$\dashv$	σ	☺
$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e}$	$\dashv$?	$a \oplus b$	☹
$\overline{a}.\mathsf{e}$	$\dashv$?	$a \oplus 0$	☹
0	$\dashv$?	σ	

Compliance, examples

$\overline{a}.\mathsf{e}$	$\dashv$	a	☺
$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e}$	$\dashv$	$a + b$	☺
e	$\dashv$	σ	☺
$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e}$	$\dashv$?	$a \oplus b$	☹
$\overline{a}.\mathsf{e}$	$\dashv$?	$a \oplus 0$	☹
0	$\dashv$?	σ	☹

Subcontract, formally

Set-theoretic interpretation of contracts

$$[\![\sigma]\!]^{\mathsf{s}} \stackrel{\text{def}}{=} \{\rho \mid \rho \dashv \sigma\}$$

Subcontract

$$\sigma \sqsubseteq \tau \stackrel{\text{def}}{\iff} [\![\sigma]\!]^{\mathsf{s}} \subseteq [\![\tau]\!]^{\mathsf{s}}$$

$$\simeq \; = \; \sqsubseteq \cap \sqsupseteq$$

Subcontract, formally

Set-theoretic interpretation of contracts

$$[\![\sigma]\!]^{\mathsf{s}} \stackrel{\text{def}}{=} \{\rho \mid \rho \dashv \sigma\}$$

Subcontract

$$\sigma \sqsubseteq \tau \stackrel{\text{def}}{\iff} [\![\sigma]\!]^{\mathsf{s}} \subseteq [\![\tau]\!]^{\mathsf{s}}$$

$$\simeq \; = \; \sqsubseteq \cap \sqsupseteq$$

Subcontract, (counter)examples

$\overline{a}.e$	$a + b$	$\not\sqsubseteq$	$a \oplus b$
$\overline{a}.e$	a	$\not\sqsubseteq$	0
$e + a$	0	$\not\sqsubseteq$	$\overline{a}$

Reduction of nondeterminism

$$\sigma \oplus \tau \sqsubseteq \sigma$$

Subcontract, (counter)examples

$\overline{a}.\mathsf{e}$		$a + b \not\sqsubseteq a \oplus b$
$\overline{a}.\mathsf{e}$		$a \not\sqsubseteq 0$
$\mathsf{e} + a$		$0 \not\sqsubseteq \overline{a}$

Reduction of nondeterminism

$$\sigma \oplus \tau \sqsubseteq \sigma$$

Properties of strong subcontract

Internal choice = intersection

$$[\![\sigma \oplus \tau]\!]^{\mathsf{s}} = [\![\sigma]\!]^{\mathsf{s}} \cap [\![\tau]\!]^{\mathsf{s}}$$

External choice $\neq$ union

- there are clients in $[\![a + b]\!]^{\mathsf{s}}$ that are not in $[\![a]\!]^{\mathsf{s}} \cup [\![b]\!]^{\mathsf{s}}$:

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \in [\![a + b]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \notin [\![a]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \notin [\![b]\!]^{\mathsf{s}}$$

- sometimes $+$ is $\oplus$ in disguise:

$$\alpha.\sigma + \alpha.\tau \simeq \alpha.(\sigma \oplus \tau)$$

- interferences:

$$\overline{a}.\mathsf{e} + \overline{b} \in [\![a]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} + \overline{b} \notin [\![a + b]\!]^{\mathsf{s}}$$

Properties of strong subcontract

Internal choice = intersection

$$[\![\sigma \oplus \tau]\!]^{\mathsf{s}} = [\![\sigma]\!]^{\mathsf{s}} \cap [\![\tau]\!]^{\mathsf{s}}$$

External choice $\neq$ union

- there are clients in $[\![a + b]\!]^{\mathsf{s}}$ that are not in $[\![a]\!]^{\mathsf{s}} \cup [\![b]\!]^{\mathsf{s}}$:

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \in [\![a + b]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \notin [\![a]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \notin [\![b]\!]^{\mathsf{s}}$$

- sometimes $+$ is $\oplus$ in disguise:

$$\alpha.\sigma + \alpha.\tau \simeq \alpha.(\sigma \oplus \tau)$$

- interferences:

$$\overline{a}.\mathsf{e} + \overline{b} \in [\![a]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} + \overline{b} \notin [\![a + b]\!]^{\mathsf{s}}$$

Properties of strong subcontract

Internal choice = intersection

$$[\![\sigma \oplus \tau]\!]^{\mathsf{s}} = [\![\sigma]\!]^{\mathsf{s}} \cap [\![\tau]\!]^{\mathsf{s}}$$

External choice $\neq$ union

- there are clients in $[\![a + b]\!]^{\mathsf{s}}$ that are not in $[\![a]\!]^{\mathsf{s}} \cup [\![b]\!]^{\mathsf{s}}$:

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \in [\![a + b]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \notin [\![a]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \notin [\![b]\!]^{\mathsf{s}}$$

- sometimes $+$ is $\oplus$ in disguise:

$$\alpha.\sigma + \alpha.\tau \simeq \alpha.(\sigma \oplus \tau)$$

- interferences:

$$\overline{a}.\mathsf{e} + \overline{b} \in [\![a]\!]^{\mathsf{s}} \qquad \overline{a}.\mathsf{e} + \overline{b} \notin [\![a + b]\!]^{\mathsf{s}}$$

Properties of strong subcontract

Internal choice = intersection

$$[\![\sigma \oplus \tau]\!]^{s} = [\![\sigma]\!]^{s} \cap [\![\tau]\!]^{s}$$

External choice ≠ union

- there are clients in $[\![a + b]\!]^{s}$ that are not in $[\![a]\!]^{s} \cup [\![b]\!]^{s}$:

$$\overline{a}.e \oplus \overline{b}.e \in [\![a + b]\!]^{s} \qquad \overline{a}.e \oplus \overline{b}.e \notin [\![a]\!]^{s} \qquad \overline{a}.e \oplus \overline{b}.e \notin [\![b]\!]^{s}$$

- sometimes $+$ is $\oplus$ in disguise:

$$\alpha.\sigma + \alpha.\tau \simeq \alpha.(\sigma \oplus \tau)$$

- interferences:

$$\overline{a}.e + \overline{b} \in [\![a]\!]^{s} \qquad \overline{a}.e + \overline{b} \notin [\![a + b]\!]^{s}$$

Properties of strong subcontract

Proposition

$\sqsubseteq$ *is a precongruence*

$$\sigma \sqsubseteq \tau \qquad \Rightarrow \qquad \begin{cases} \alpha.\sigma \sqsubseteq \alpha.\tau \\ \sigma \oplus \sigma' \sqsubseteq \tau \oplus \sigma' \\ \sigma + \sigma' \sqsubseteq \tau + \sigma' \end{cases}$$

+ nice axiomatization
+ can be used for safe replacement of parts of services

Strong subcontract: axioms

$$
\begin{array}{lrcl}
\text{(e1)} & \sigma + \sigma & = & \sigma \\
\text{(e2)} & \sigma + \tau & = & \tau + \sigma \\
\text{(e3)} & \sigma + (\sigma' + \sigma'') & = & (\sigma + \sigma') + \sigma'' \\
\text{(e4)} & \sigma + 0 & = & \sigma \\
\\
\text{(i1)} & \sigma \oplus \sigma & = & \sigma \\
\text{(i2)} & \sigma \oplus \tau & = & \tau \oplus \sigma \\
\text{(i3)} & \sigma \oplus (\sigma' \oplus \sigma'') & = & (\sigma \oplus \sigma') \oplus \sigma'' \\
\\
\text{(d1)} & \sigma + (\sigma' \oplus \sigma'') & = & (\sigma + \sigma') \oplus (\sigma + \sigma'') \\
\text{(d2)} & \sigma \oplus (\sigma' + \sigma'') & = & (\sigma \oplus \sigma') + (\sigma \oplus \sigma'') \\
\text{(d3)} & \alpha.\sigma + \alpha.\tau & = & \alpha.(\sigma \oplus \tau) \\
\text{(d4)} & \alpha.\sigma \oplus \alpha.\tau & = & \alpha.(\sigma \oplus \tau) \\
\\
\text{(red)} & \sigma \oplus \tau & \leq & \sigma
\end{array}
$$

Orchestrators

Limitations of $\sqsubseteq$

$\sqsubseteq$ does not support extensions. . .

$$a \not\sqsubseteq a + b$$

. . . because extra actions may cause interferences

$$\overline{a}.\mathsf{e} + \overline{b} \dashv a \qquad\qquad \overline{a}.\mathsf{e} + \overline{b} \not\dashv a + b$$

Why not all failures are equal

Failure due to client nondeterminism

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \parallel a \longrightarrow \overline{b}.\mathsf{e} \parallel a$$

Failure due to service nondeterminism

$$\overline{a}.\mathsf{e} \parallel a \oplus b \longrightarrow \overline{a}.\mathsf{e} \parallel a$$

Failure due to "system" nondeterminism

$$\overline{a}.\mathsf{e} + \overline{b}.c.\mathsf{e} \parallel a + b.\overline{d} \longrightarrow c.\mathsf{e} \parallel \overline{d}$$

$$\overline{a}.\mathsf{e} + \overline{b}.c.\mathsf{e} \parallel a + b.\overline{d} \longrightarrow \mathsf{e} \parallel 0$$

Why not all failures are equal

Failure due to client nondeterminism

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \parallel a \longrightarrow \overline{b}.\mathsf{e} \parallel a$$

Failure due to service nondeterminism

$$\overline{a}.\mathsf{e} \parallel a \oplus b \longrightarrow \overline{a}.\mathsf{e} \parallel a$$

Failure due to “system” nondeterminism

$$\overline{a}.\mathsf{e} + \overline{b}.c.\mathsf{e} \parallel a + b.\overline{d} \longrightarrow c.\mathsf{e} \parallel \overline{d}$$

$$\overline{a}.\mathsf{e} + \overline{b}.c.\mathsf{e} \parallel a + b.\overline{d} \longrightarrow \mathsf{e} \parallel 0$$

Why not all failures are equal

Failure due to client nondeterminism

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \parallel a \longrightarrow \overline{b}.\mathsf{e} \parallel a$$

Failure due to service nondeterminism

$$\overline{a}.\mathsf{e} \parallel a \oplus b \longrightarrow \overline{a}.\mathsf{e} \parallel a$$

Failure due to "system" nondeterminism

$$\overline{a}.\mathsf{e} + \overline{b}.c.\mathsf{e} \parallel a + b.\overline{d} \longrightarrow c.\mathsf{e} \parallel \overline{d}$$

$$\overline{a}.\mathsf{e} + \overline{b}.c.\mathsf{e} \parallel a + b.\overline{d} \longrightarrow \mathsf{e} \parallel 0$$

Why not all failures are equal

Failure due to client nondeterminism

$$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \parallel a \longrightarrow \overline{b}.\mathsf{e} \parallel a$$

Failure due to service nondeterminism

$$\overline{a}.\mathsf{e} \parallel a \oplus b \longrightarrow \overline{a}.\mathsf{e} \parallel a$$

Failure due to "system" nondeterminism

$$\overline{a}.\mathsf{e} + \overline{b}.c.\mathsf{e} \parallel a + b.\overline{d} \longrightarrow c.\mathsf{e} \parallel \overline{d}$$

$$\overline{a}.\mathsf{e} + \overline{b}.c.\mathsf{e} \parallel a + b.\overline{d} \longrightarrow \mathsf{e} \parallel 0$$

Idea: orchestrated interaction

client	⇆	orchestrator	⇆	service
$\overline{a}$	$\rightarrow$	$\langle a, \overline{a} \rangle$	$\rightarrow$	a
a	$\leftarrow$	$\langle \overline{a}, a \rangle$	$\leftarrow$	$\overline{a}$
$\overline{\alpha}$				α
$\overline{\alpha}$		$\langle \alpha, \overline{\alpha} \rangle$		

Idea: orchestrated interaction

client	$\leftrightarrows$	orchestrator	$\leftrightarrows$	service
$\overline{a}$	$\rightarrow$	$\langle a, \overline{a} \rangle$	$\rightarrow$	a
a	$\leftarrow$	$\langle \overline{a}, a \rangle$	$\leftarrow$	$\overline{a}$
$\overline{\alpha}$				α
$\overline{\alpha}$		$\langle \alpha, \overline{\alpha} \rangle$		

Idea: orchestrated interaction

client	$\leftrightarrows$	orchestrator	$\leftrightarrows$	service
$\overline{a}$	$\rightarrow$	$\langle a, \overline{a} \rangle$	$\rightarrow$	a
a	$\leftarrow$	$\langle \overline{a}, a \rangle$	$\leftarrow$	$\overline{a}$
$\overline{\alpha}$				α
$\overline{\alpha}$		$\langle \alpha, \overline{\alpha} \rangle$		

Idea: orchestrated interaction

client	$\leftrightarrows$	orchestrator	$\leftrightarrows$	service
$\overline{a}$	$\rightarrow$	$\langle a, \overline{a} \rangle$	$\rightarrow$	a
a	$\leftarrow$	$\langle \overline{a}, a \rangle$	$\leftarrow$	$\overline{a}$
$\overline{\alpha}$				α
$\overline{\alpha}$		$\langle \alpha, \overline{\alpha} \rangle$		

Idea: orchestrated interaction

client	$\leftrightarrows$	orchestrator	$\leftrightarrows$	service
$\overline{a}$	$\rightarrow$	$\langle a, \overline{a} \rangle$	$\rightarrow$	a
a	$\leftarrow$	$\langle \overline{a}, a \rangle$	$\leftarrow$	$\overline{a}$
$\overline{\alpha}$				α
$\overline{\alpha}$		$\langle \alpha, \overline{\alpha} \rangle$		

Synchronous orchestrators

f	$::=$		**orchestrator**
		0	(null)
	$\mid$	$\mu.f$	(action prefix)
	$\mid$	$f \vee f$	(disjunction)

μ	$::=$		**orchestration action**
		$\langle a, \overline{a} \rangle$	(input/output)
	$\mid$	$\langle \overline{a}, a \rangle$	(output/input)

Orchestrator semantics

Orchestrator transitions

$$\mu.f \xrightarrow{\mu} f \qquad \frac{f \xrightarrow{\mu} f'}{f \vee g \xrightarrow{\mu} f'}$$

Trace semantics for orchestrators

$$[\![f]\!] \stackrel{\text{def}}{=} \{\mu_1 \cdots \mu_n \mid \exists g : f \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} g\}$$

Orchestrated compliance, formally

Orchestrated systems

$$\rho \parallel_f \sigma$$

Orchestrated system transitions

$$\frac{\rho \longrightarrow \rho'}{\rho \parallel_f \sigma \longrightarrow \rho' \parallel_f \sigma} \qquad \frac{\sigma \longrightarrow \sigma'}{\rho \parallel_f \sigma \longrightarrow \rho \parallel_f \sigma'}$$

$$\frac{\rho \xrightarrow{\overline{\alpha}} \rho' \quad f \xrightarrow{\langle \alpha, \overline{\alpha} \rangle} f' \quad \sigma \xrightarrow{\alpha} \sigma'}{\rho \parallel_f \sigma \longrightarrow \rho' \parallel_{f'} \sigma'}$$

Weak compliance

$$f : \rho \dashv\!\!\!\vdots\; \sigma \stackrel{\text{def}}{\iff} \rho \parallel_f \sigma \Longrightarrow \rho' \parallel_{f'} \sigma' \not\longrightarrow \text{ implies } \rho' \xrightarrow{\mathsf{e}}$$

Weak compliance, examples

$$\langle a, \overline{a} \rangle \quad : \quad \overline{a}.\mathsf{e} \ \dashv\!\!\!\mid\!\mid \ ? \ \ a + b$$

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle \quad : \quad \overline{a}.\mathsf{e} \ \dashv\!\!\!\mid\!\mid \ a + b$$

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle \quad : \quad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \ \dashv\!\!\!\mid\!\mid \ a + b$$

$$0 \quad : \quad \mathsf{e} + a \ \dashv\!\!\!\mid\!\mid \ \overline{a}$$

$$f \quad : \quad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \ \dashv\!\!\!\mid\!\mid \ a \oplus b$$

Weak compliance, examples

$$\langle a, \overline{a} \rangle \quad : \quad \overline{a}.\mathsf{e} \;\dashv\!\!\!\!\!\;\mid\mid\; a + b \qquad ☺$$

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle \quad : \quad \overline{a}.\mathsf{e} \;\dashv\!\!\!\!\!\;\mid\mid\; ? \; a + b$$

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle \quad : \quad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \;\dashv\!\!\!\!\!\;\mid\mid\; a + b$$

$$0 \quad : \quad \mathsf{e} + a \;\dashv\!\!\!\!\!\;\mid\mid\; \overline{a}$$

$$f \quad : \quad \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \;\dashv\!\!\!\!\!\;\mid\mid\; a \oplus b$$

Weak compliance, examples

$$
\begin{array}{rcrcll}
\langle a, \overline{a} \rangle & : & \overline{a}.\mathsf{e} & \dashv\!\!\vdots & a + b & ☺ \\
\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle & : & \overline{a}.\mathsf{e} & \dashv\!\!\vdots & a + b & ☺ \\
\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle & : & \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} & \dashv\!\!\vdots\ ? & a + b & \\
0 & : & \mathsf{e} + a & \dashv\!\!\vdots & \overline{a} & \\
f & : & \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} & \dashv\!\!\vdots & a \oplus b &
\end{array}
$$

Weak compliance, examples

$$
\begin{array}{rcrcll}
\langle a, \overline{a} \rangle & : & \overline{a}.\mathsf{e} & \dashv & a + b & ☺ \\
\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle & : & \overline{a}.\mathsf{e} & \dashv & a + b & ☺ \\
\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle & : & \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} & \dashv & a + b & ☺ \\
0 & : & \mathsf{e} + a & \dashv ? & \overline{a} & \\
f & : & \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} & \dashv & a \oplus b &
\end{array}
$$

Weak compliance, examples

$$\langle a, \overline{a} \rangle \; : \; \overline{a}.\mathsf{e} \dashv\!\!\!\mid\; a + b \quad ☺$$

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle \; : \; \overline{a}.\mathsf{e} \dashv\!\!\!\mid\; a + b \quad ☺$$

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle \; : \; \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \dashv\!\!\!\mid\; a + b \quad ☺$$

$$0 \; : \; \mathsf{e} + a \dashv\!\!\!\mid\; \overline{a} \quad ☺$$

$$f \; : \; \overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e} \dashv\!\!\!\mid\, ?\; a \oplus b$$

Weak compliance, examples

$\langle a, \overline{a} \rangle$	:	$\overline{a}.\mathsf{e}$	$\dashv\!\!\!\!\vdots$	$a + b$	☺
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	:	$\overline{a}.\mathsf{e}$	$\dashv\!\!\!\!\vdots$	$a + b$	☺
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	:	$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e}$	$\dashv\!\!\!\!\vdots$	$a + b$	☺
0	:	$\mathsf{e} + a$	$\dashv\!\!\!\!\vdots$	$\overline{a}$	☺
f	:	$\overline{a}.\mathsf{e} \oplus \overline{b}.\mathsf{e}$	$\dashv\!\!\!\!\vdots$	$a \oplus b$	☹

Weak subcontract, formally

Set-theoretic interpretation of contracts

$$[\![\sigma]\!]^{\mathsf{w}} \stackrel{\text{def}}{=} \{\rho \mid \exists f : f : \rho \dashv\!\!\! \mid\mid \sigma\}$$

Weak subcontract

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} [\![\sigma]\!]^{\mathsf{s}} \subseteq [\![\tau]\!]^{\mathsf{w}}$$

A few doubts. . .

- is $\preceq$ the subcontract relation we're looking for?
- is $\preceq$ a preorder?

Weak subcontract, formally

Set-theoretic interpretation of contracts

$$[\![\sigma]\!]^{\mathsf{w}} \stackrel{\text{def}}{=} \{\rho \mid \exists f : f : \rho \dashv\!\!\!\!\!\!\;\;\; \sigma\}$$

Weak subcontract

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} [\![\sigma]\!]^{\mathsf{s}} \subseteq [\![\tau]\!]^{\mathsf{w}}$$

A few doubts. . .

- is $\preceq$ the subcontract relation we're looking for?
- is $\preceq$ a preorder?

Weak subcontract, formally

Set-theoretic interpretation of contracts

$$[\![\sigma]\!]^{\mathsf{w}} \stackrel{\text{def}}{=} \{\rho \mid \exists f : f : \rho \dashv\!\!\!\mid \sigma\}$$

Weak subcontract

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} [\![\sigma]\!]^{\mathsf{s}} \subseteq [\![\tau]\!]^{\mathsf{w}}$$

A few doubts...

- is $\preceq$ the subcontract relation we're looking for?
- is $\preceq$ a preorder?

Universal orchestrators

$$\sigma \preceq \tau \iff \text{for every } \rho, \rho \dashv \sigma \text{ implies } f : \rho \dashv\!\!\mid \tau \text{ for some } f$$

Universal orchestrator

$$f : \sigma \preceq \tau \stackrel{\text{def}}{\iff} \text{for every } \rho, \rho \dashv \sigma \text{ implies } f : \rho \dashv\!\!\mid \tau$$

f is the *universal orchestrator* for $\sigma \preceq \tau$

Proposition (existence of universal orchestrator)

$\sigma \preceq \tau$ if and only if $f : \sigma \preceq \tau$ for some orchestrator f

Universal orchestrators

$$\sigma \preceq \tau \iff \text{for every } \rho, \rho \dashv \sigma \text{ implies } f : \rho \dashv\!\!\!\mid \tau \text{ for some } f$$

Universal orchestrator

$$f : \sigma \preceq \tau \stackrel{\text{def}}{\iff} \text{for every } \rho, \rho \dashv \sigma \text{ implies } f : \rho \dashv\!\!\!\mid \tau$$

f is the *universal orchestrator* for $\sigma \preceq \tau$

Proposition (existence of universal orchestrator)

$\sigma \preceq \tau$ if and only if $f : \sigma \preceq \tau$ for some orchestrator f

Universal orchestrators

$$\sigma \preceq \tau \iff \text{for every } \rho, \rho \dashv \sigma \text{ implies } f : \rho \dashv\!\!\!\!\,\vdots\; \tau \text{ for some } f$$

Universal orchestrator

$$f : \sigma \preceq \tau \stackrel{\text{def}}{\iff} \text{for every } \rho, \rho \dashv \sigma \text{ implies } f : \rho \dashv\!\!\!\!\,\vdots\; \tau$$

f is the *universal orchestrator* for $\sigma \preceq \tau$

Proposition (existence of universal orchestrator)

$\sigma \preceq \tau$ if and only if $f : \sigma \preceq \tau$ for some orchestrator f

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a + b$	$a + b$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a \oplus b$	$a \oplus b$
$\langle a, \overline{a} \rangle$	$a + b$	a
$\langle a, \overline{a} \rangle$	$a \oplus b$	$a \oplus 0$
0	σ	0

Theorem

$f : \rho \dashv\!| \, \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \overline{a}\rangle \vee \langle b, \overline{b}\rangle$	$a + b$	$a + b$
$\langle a, \overline{a}\rangle \vee \langle b, \overline{b}\rangle$	$a \oplus b$	$a \oplus b$
$\langle a, \overline{a}\rangle$	$a + b$	a
$\langle a, \overline{a}\rangle$	$a \oplus b$	$a \oplus 0$
0	σ	0

Theorem

$f : \rho \dashv\!\!| \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a + b$	$a + b$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a \oplus b$	$a \oplus b$
$\langle a, \overline{a} \rangle$	$a + b$	a
$\langle a, \overline{a} \rangle$	$a \oplus b$	$a \oplus 0$
0	σ	0

Theorem

$f : \rho \dashv\mid \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a + b$	$a + b$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a \oplus b$	$a \oplus b$
$\langle a, \overline{a} \rangle$	$a + b$	a
$\langle a, \overline{a} \rangle$	$a \oplus b$	$a \oplus 0$
0	σ	0

Theorem

$f : \rho \dashv\!| \, \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a + b$	$a + b$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a \oplus b$	$a \oplus b$
$\langle a, \overline{a} \rangle$	$a + b$	a
$\langle a, \overline{a} \rangle$	$a \oplus b$	$a \oplus 0$
0	σ	0

Theorem

$f : \rho \dashv\!| \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a + b$	$a + b$
$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle$	$a \oplus b$	$a \oplus b$
$\langle a, \overline{a} \rangle$	$a + b$	a
$\langle a, \overline{a} \rangle$	$a \oplus b$	$a \oplus 0$
0	σ	0

Theorem

$f : \rho \dashv\!\mid \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Weak subcontract, examples

Reduction of nondeterminism ($\preceq$ embeds $\sqsubseteq$)

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle : a \oplus b \preceq a$$
$$\Updownarrow$$
$$a \oplus b \sqsubseteq (\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle)(a) = a$$

Width extension

$$\langle a, \overline{a} \rangle : a \preceq a + b$$
$$\Updownarrow$$
$$a \sqsubseteq \langle a, \overline{a} \rangle (a + b) = a$$

Depth extension

$$0 : 0 \preceq \sigma$$
$$\Updownarrow$$
$$0 \sqsubseteq 0(\sigma) = 0$$

Weak subcontract, examples

Reduction of nondeterminism ($\preceq$ embeds $\sqsubseteq$)

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle : a \oplus b \preceq a$$
$$\Updownarrow$$
$$a \oplus b \sqsubseteq (\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle)(a) = a$$

Width extension

$$\langle a, \overline{a} \rangle : a \preceq a + b$$
$$\Updownarrow$$
$$a \sqsubseteq \langle a, \overline{a} \rangle (a + b) = a$$

Depth extension

$$0 : 0 \preceq \sigma$$
$$\Updownarrow$$
$$0 \sqsubseteq 0(\sigma) = 0$$

Weak subcontract, examples

Reduction of nondeterminism ($\preceq$ embeds $\sqsubseteq$)

$$\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle : a \oplus b \preceq a$$
$$\Updownarrow$$
$$a \oplus b \sqsubseteq (\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle)(a) = a$$

Width extension

$$\langle a, \overline{a} \rangle : a \preceq a + b$$
$$\Updownarrow$$
$$a \sqsubseteq \langle a, \overline{a} \rangle (a + b) = a$$

Depth extension

$$0 : 0 \preceq \sigma$$
$$\Updownarrow$$
$$0 \sqsubseteq 0(\sigma) = 0$$

On transitivity of $\preceq$

Is $\preceq$ transitive?

$$f : \sigma \preceq \sigma' \qquad g : \sigma' \preceq \tau \qquad \stackrel{?}{\Longrightarrow} \qquad h : \sigma \preceq \tau$$

$$\begin{array}{rcrcl} \langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle . \langle c, \overline{c} \rangle & : & a \oplus b.c & \preceq & a \\ \langle a, \overline{a} \rangle & : & a & \preceq & a + b.d \\ ??? & : & a \oplus b.c & \preceq & a + b.d \end{array}$$

Proposition (Orchestrator application is monotone)

$\sigma \sqsubseteq \tau$ *implies* $f(\sigma) \sqsubseteq f(\tau)$

$$\sigma \sqsubseteq f(\sigma') \qquad \sigma' \sqsubseteq g(\tau) \qquad \Longrightarrow \qquad \sigma \sqsubseteq f(g(\tau))$$

$\preceq$ is transitive if $f \circ g$ is an orchestrator

On transitivity of $\preceq$

Is $\preceq$ transitive?

$$f : \sigma \preceq \sigma' \qquad g : \sigma' \preceq \tau \qquad \stackrel{?}{\Longrightarrow} \qquad h : \sigma \preceq \tau$$

$$\begin{array}{rcrcl} \langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle . \langle c, \overline{c} \rangle & : & a \oplus b.c & \preceq & a \\ \langle a, \overline{a} \rangle & : & a & \preceq & a + b.d \\ ??? & : & a \oplus b.c & \preceq & a + b.d \end{array}$$

Proposition (Orchestrator application is monotone)

$\sigma \sqsubseteq \tau$ *implies* $f(\sigma) \sqsubseteq f(\tau)$

$$\sigma \sqsubseteq f(\sigma') \qquad \sigma' \sqsubseteq g(\tau) \qquad \Longrightarrow \qquad \sigma \sqsubseteq f(g(\tau))$$

$\preceq$ **is transitive if** $f \circ g$ **is an orchestrator**

On transitivity of $\preceq$

Is $\preceq$ transitive?

$$f : \sigma \preceq \sigma' \qquad g : \sigma' \preceq \tau \qquad \stackrel{?}{\Longrightarrow} \qquad h : \sigma \preceq \tau$$

$$\begin{array}{rcrcl} \langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle.\langle c, \overline{c} \rangle & : & a \oplus b.c & \preceq & a \\ \langle a, \overline{a} \rangle & : & a & \preceq & a + b.d \\ ??? & : & a \oplus b.c & \preceq & a + b.d \end{array}$$

Proposition (Orchestrator application is monotone)

$\sigma \sqsubseteq \tau$ *implies* $f(\sigma) \sqsubseteq f(\tau)$

$$\sigma \sqsubseteq f(\sigma') \qquad \sigma' \sqsubseteq g(\tau) \qquad \Longrightarrow \qquad \sigma \sqsubseteq f(g(\tau))$$

$\preceq$ is transitive if $f \circ g$ is an orchestrator

On transitivity of $\preceq$

Is $\preceq$ transitive?

$$f : \sigma \preceq \sigma' \qquad g : \sigma' \preceq \tau \qquad \stackrel{?}{\Longrightarrow} \qquad h : \sigma \preceq \tau$$

$$\begin{array}{rcrcl} \langle a, \overline{a}\rangle \vee \langle b, \overline{b}\rangle.\langle c, \overline{c}\rangle & : & a \oplus b.c & \preceq & a \\ \langle a, \overline{a}\rangle & : & a & \preceq & a + b.d \\ ??? & : & a \oplus b.c & \preceq & a + b.d \end{array}$$

Proposition (Orchestrator application is monotone)

$\sigma \sqsubseteq \tau$ *implies* $f(\sigma) \sqsubseteq f(\tau)$

$$\sigma \sqsubseteq f(\sigma') \qquad \sigma' \sqsubseteq g(\tau) \qquad \Longrightarrow \qquad \sigma \sqsubseteq f(g(\tau))$$

$\preceq$ is transitive if $f \circ g$ is an orchestrator

Orchestrator composition and transitivity of $\preceq$

Orchestrator composition

$$f \wedge g$$

- $f \wedge g$ permits the traces permitted by f and by g
- $f \wedge g$ forbids the traces forbidden by either f or by g

$$[\![f \wedge g]\!] \stackrel{\text{def}}{=} [\![f]\!] \cap [\![g]\!]$$

Proposition

$f(g(\sigma)) \simeq (f \wedge g)(\sigma)$

Orchestrator composition and transitivity of $\preceq$

Orchestrator composition

$$f \wedge g$$

- $f \wedge g$ permits the traces permitted by f and by g
- $f \wedge g$ forbids the traces forbidden by either f or by g

$$[\![f \wedge g]\!] \stackrel{\mathrm{def}}{=} [\![f]\!] \cap [\![g]\!]$$

Proposition

$f(g(\sigma)) \simeq (f \wedge g)(\sigma)$

Towards a deduction system for $\preceq$

Problem: $\preceq$ is not a precongruence w.r.t. $+$

$$\begin{array}{ccccccc} & & 0 & \preceq & & & a \\ a.b & & & \preceq & a.b & & \\ a.b & + & 0 & \not\preceq & a.b & + & a \simeq a.(b \oplus 0) \end{array}$$

Proposition (distributivity of orchestration application)

1. $f(\sigma) + f(\tau) \simeq f(\sigma + \tau)$
2. $f(\sigma) \oplus f(\tau) \simeq f(\sigma \oplus \tau)$

Corollary

$f : \sigma_1 \preceq \tau_1$ *and* $f : \sigma_2 \preceq \tau_2$ *implies* $f : \sigma_1 + \sigma_2 \preceq \tau_1 + \tau_2$

- precongruence is granted if the orchestrator is oblivious of the particular branch taken

Towards a deduction system for $\preceq$

Problem: $\preceq$ is not a precongruence w.r.t. $+$

$$
\begin{array}{ccccccc}
 & & 0 & \preceq & & & a \\
a.b & & & \preceq & a.b & & \\
a.b & + & 0 & \not\preceq & a.b & + & a \simeq a.(b \oplus 0)
\end{array}
$$

Proposition (distributivity of orchestration application)

1. $f(\sigma) + f(\tau) \simeq f(\sigma + \tau)$
2. $f(\sigma) \oplus f(\tau) \simeq f(\sigma \oplus \tau)$

Corollary

$f : \sigma_1 \preceq \tau_1$ *and* $f : \sigma_2 \preceq \tau_2$ *implies* $f : \sigma_1 + \sigma_2 \preceq \tau_1 + \tau_2$

- precongruence is granted if the orchestrator is oblivious of the particular branch taken

Towards a deduction system for $\preceq$

Problem: $\preceq$ is not a precongruence w.r.t. $+$

$$\begin{array}{ccccccc} & & 0 & \preceq & & & a \\ a.b & & & \preceq & a.b & & \\ a.b & + & 0 & \not\preceq & a.b & + & a \simeq a.(b \oplus 0) \end{array}$$

Proposition (distributivity of orchestration application)

1. $f(\sigma) + f(\tau) \simeq f(\sigma + \tau)$
2. $f(\sigma) \oplus f(\tau) \simeq f(\sigma \oplus \tau)$

Corollary

$f : \sigma_1 \preceq \tau_1$ *and* $f : \sigma_2 \preceq \tau_2$ *implies* $f : \sigma_1 + \sigma_2 \preceq \tau_1 + \tau_2$

- precongruence is granted if the orchestrator is oblivious of the particular branch taken

Deduction system for $\preceq$

(red) $$I(\sigma) : \sigma \oplus \tau \leq \sigma$$

(prefix) $$\frac{f : \sigma \leq \tau}{\langle \alpha, \overline{\alpha} \rangle . f : \alpha.\sigma \leq \alpha.\tau}$$

(width) $$\frac{[\![I(\sigma) \wedge I(\tau)]\!] = \{\varepsilon\}}{I(\sigma) : \sigma \leq \sigma + \tau}$$

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

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

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

The deduction system is *sound* and *complete*

$$f : \sigma \preceq \tau \iff f : \sigma \leq \tau$$

- completeness regards orchestrators too

Deduction system for $\preceq$

(red) $$I(\sigma) : \sigma \oplus \tau \leq \sigma$$

(prefix) $$\frac{f : \sigma \leq \tau}{\langle \alpha, \overline{\alpha} \rangle.f : \alpha.\sigma \leq \alpha.\tau}$$

(width) $$\frac{[\![I(\sigma) \wedge I(\tau)]\!] = \{\varepsilon\}}{I(\sigma) : \sigma \leq \sigma + \tau}$$

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

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

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

The deduction system is *sound* and *complete*

$$f : \sigma \preceq \tau \iff f : \sigma \leq \tau$$

- completeness regards orchestrators too

Interpretations of orchestrators

As **mediators**

$$\rho \parallel_f \sigma$$

As **morphisms/behavioral coercions**

$$f : \sigma \preceq \tau \iff \sigma \sqsubseteq f(\tau) \qquad f : \tau \to \sigma$$

As **assumptions** on the environment

$$\langle a, \overline{a} \rangle : a \preceq a + b$$

- it is safe to replace a with $a + b$ if nobody ever tries to perform $\overline{b}$

Interpretations of orchestrators

As **mediators**

$$\rho \parallel_f \sigma$$

As **morphisms/behavioral coercions**

$$f : \sigma \preceq \tau \iff \sigma \sqsubseteq f(\tau) \qquad\qquad f : \tau \to \sigma$$

As **assumptions** on the environment

$$\langle a, \overline{a} \rangle : a \preceq a + b$$

- it is safe to replace a with $a + b$ if nobody ever tries to perform $\overline{b}$

Interpretations of orchestrators

As **mediators**

$$\rho \parallel_f \sigma$$

As **morphisms/behavioral coercions**

$$f : \sigma \preceq \tau \iff \sigma \sqsubseteq f(\tau) \qquad\qquad f : \tau \to \sigma$$

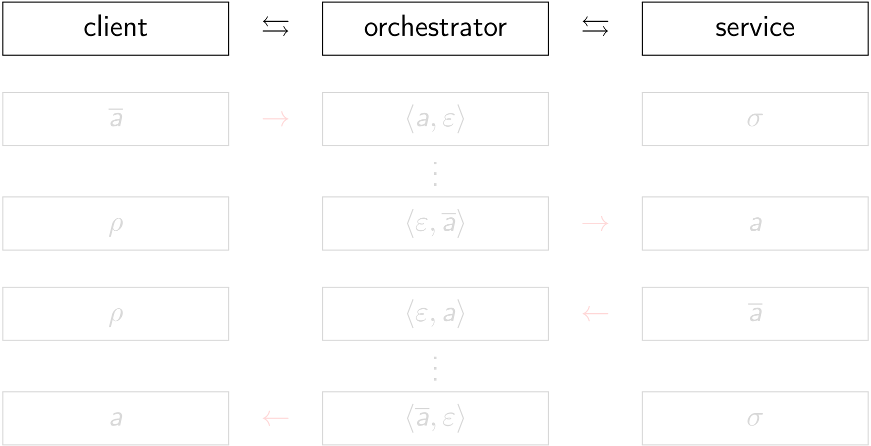
As **assumptions** on the environment

$$\langle a, \overline{a} \rangle : a \preceq a + b$$

- it is safe to replace a with $a + b$ if nobody ever tries to perform $\overline{b}$

Buffered orchestrators

Buffered orchestrators



Buffered orchestrators

client	$\leftrightarrows$	orchestrator	$\leftrightarrows$	service
$\overline{a}$	$\rightarrow$	$\langle a, \varepsilon \rangle$		σ
		$\vdots$		
ρ		$\langle \varepsilon, \overline{a} \rangle$	$\rightarrow$	a
ρ		$\langle \varepsilon, a \rangle$	$\leftarrow$	$\overline{a}$
		$\vdots$		
a	$\leftarrow$	$\langle \overline{a}, \varepsilon \rangle$		σ

Buffered orchestrators

client	$\leftrightarrows$	orchestrator	$\leftrightarrows$	service
$\overline{a}$	$\rightarrow$	$\langle a, \varepsilon \rangle$		σ
		$\vdots$		
ρ		$\langle \varepsilon, \overline{a} \rangle$	$\rightarrow$	a
ρ		$\langle \varepsilon, a \rangle$	$\leftarrow$	$\overline{a}$
		$\vdots$		
a	$\leftarrow$	$\langle \overline{a}, \varepsilon \rangle$		σ

Syntax of buffered orchestrators

$$
\begin{array}{llll}
f & ::= & & \textbf{orchestrator} \\
 & & 0 & \text{(null)} \\
 & | & \mu.f & \text{(action prefix)} \\
 & | & f \vee f & \text{(disjunction)}
\end{array}
$$

$$
\begin{array}{llll}
\mu & ::= & & \textbf{orchestration action} \\
 & & \langle a, \overline{a} \rangle & \text{(input/output)} \\
 & | & \langle \overline{a}, a \rangle & \text{(output/input)} \\
 & | & \langle \alpha, \varepsilon \rangle & \text{(action/–)} \\
 & | & \langle \varepsilon, \alpha \rangle & \text{(–/action)}
\end{array}
$$

Valid k-orchestrators

Not every orchestrator makes sense

orchestrator	valid	rank
$\langle \varepsilon, a \rangle . \langle \overline{a}, \varepsilon \rangle$		
$\langle a, \varepsilon \rangle . \langle a, \varepsilon \rangle$		
$\langle \overline{a}, \varepsilon \rangle$		
$\langle \varepsilon, \overline{a} \rangle$		
$\langle a, \varepsilon \rangle . \langle \overline{a}, \varepsilon \rangle$		

Valid k-orchestrators are...

- directional
- finite-state
- fair

Valid k-orchestrators

Not every orchestrator makes sense

orchestrator	valid	rank
$\langle \varepsilon, a \rangle . \langle \overline{a}, \varepsilon \rangle$	☺	≥ 1
$\langle a, \varepsilon \rangle . \langle a, \varepsilon \rangle$		
$\langle \overline{a}, \varepsilon \rangle$		
$\langle \varepsilon, \overline{a} \rangle$		
$\langle a, \varepsilon \rangle . \langle \overline{a}, \varepsilon \rangle$		

Valid k-orchestrators are...

- directional
- finite-state
- fair

Valid k-orchestrators

Not every orchestrator makes sense

orchestrator	valid	rank
$\langle \varepsilon, a \rangle . \langle \overline{a}, \varepsilon \rangle$	☺	≥ 1
$\langle a, \varepsilon \rangle . \langle a, \varepsilon \rangle$	☺	≥ 2
$\langle \overline{a}, \varepsilon \rangle$		
$\langle \varepsilon, \overline{a} \rangle$		
$\langle a, \varepsilon \rangle . \langle \overline{a}, \varepsilon \rangle$		

Valid k-orchestrators are...

- directional
- finite-state
- fair

Valid *k*-orchestrators

Not every orchestrator makes sense

orchestrator	valid	rank
$\langle \varepsilon, a \rangle . \langle \overline{a}, \varepsilon \rangle$	☺	≥ 1
$\langle a, \varepsilon \rangle . \langle a, \varepsilon \rangle$	☺	≥ 2
$\langle \overline{a}, \varepsilon \rangle$	☹	
$\langle \varepsilon, \overline{a} \rangle$		
$\langle a, \varepsilon \rangle . \langle \overline{a}, \varepsilon \rangle$		

Valid *k*-orchestrators are...

- directional
- finite-state
- fair

Valid k-orchestrators

Not every orchestrator makes sense

orchestrator	valid	rank
$\langle \varepsilon, a \rangle . \langle \overline{a}, \varepsilon \rangle$	☺	≥ 1
$\langle a, \varepsilon \rangle . \langle a, \varepsilon \rangle$	☺	≥ 2
$\langle \overline{a}, \varepsilon \rangle$	☹	
$\langle \varepsilon, \overline{a} \rangle$	☹	
$\langle a, \varepsilon \rangle . \langle \overline{a}, \varepsilon \rangle$		

Valid k-orchestrators are...

- directional
- finite-state
- fair

Valid k-orchestrators

Not every orchestrator makes sense

orchestrator	valid	rank
$\langle \varepsilon, a \rangle . \langle \overline{a}, \varepsilon \rangle$	☺	≥ 1
$\langle a, \varepsilon \rangle . \langle a, \varepsilon \rangle$	☺	≥ 2
$\langle \overline{a}, \varepsilon \rangle$	☹	
$\langle \varepsilon, \overline{a} \rangle$	☹	
$\langle a, \varepsilon \rangle . \langle \overline{a}, \varepsilon \rangle$	☹	

Valid k-orchestrators are...

- directional
- finite-state
- fair

Valid k-orchestrators

Not every orchestrator makes sense

orchestrator	valid	rank
$\langle \varepsilon, a \rangle . \langle \overline{a}, \varepsilon \rangle$	☺	≥ 1
$\langle a, \varepsilon \rangle . \langle a, \varepsilon \rangle$	☺	≥ 2
$\langle \overline{a}, \varepsilon \rangle$	☹	
$\langle \varepsilon, \overline{a} \rangle$	☹	
$\langle a, \varepsilon \rangle . \langle \overline{a}, \varepsilon \rangle$	☹	

Valid k-orchestrators are...

- directional
- finite-state
- fair

Weak k-compliance, formally

Orchestrated systems

$$\rho \parallel_f \sigma$$

Orchestrated system transitions

$$\cdots$$

$$\frac{\rho \xrightarrow{\overline{\alpha}} \rho' \quad f \xrightarrow{\langle \alpha, \varepsilon \rangle} f'}{\rho \parallel_f \sigma \longrightarrow \rho' \parallel_{f'} \sigma} \qquad \frac{f \xrightarrow{\langle \varepsilon, \overline{\alpha} \rangle} f' \quad \sigma \xrightarrow{\alpha} \sigma'}{\rho \parallel_f \sigma \longrightarrow \rho \parallel_{f'} \sigma'}$$

Weak k-compliance

$$f : \rho \dashv\!\mid_k \sigma \quad \overset{\text{def}}{\iff} \quad \rho \parallel_f \sigma \Longrightarrow \rho' \parallel_{f'} \sigma' \not\longrightarrow \text{ implies } \rho' \xrightarrow{\mathsf{e}}$$

Weak k-compliance, an example

$$\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle.\langle \varepsilon, b\rangle.\langle \varepsilon, a\rangle.\langle \overline{c}, c\rangle : \overline{a}.\overline{b}.c.\mathsf{e} \dashv\!\!\shortmid_1 b.a.\overline{c}$$

Weak k-subcontract, formally

Set-theoretic interpretation of contracts

$$[\![\sigma]\!]^{\mathsf{w}}_k \stackrel{\text{def}}{=} \{\rho \mid \exists f : f : \rho \dashv\!\!\vert_k \sigma\}$$

Weak subcontract

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} [\![\sigma]\!]^{\mathsf{s}} \subseteq [\![\tau]\!]^{\mathsf{w}}_k$$

Proposition (existence of universal orchestrator)

$\sigma \preceq_k \tau$ if and only if $f : \sigma \preceq_k \tau$ for some k-orchestrator f

Weak k-subcontract, formally

Set-theoretic interpretation of contracts

$$[\![\sigma]\!]^{\mathsf{w}}_{k} \stackrel{\text{def}}{=} \{\rho \mid \exists f : f : \rho \dashv\!\!\mid_k \sigma\}$$

Weak subcontract

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} [\![\sigma]\!]^{\mathsf{s}} \subseteq [\![\tau]\!]^{\mathsf{w}}_{k}$$

Proposition (existence of universal orchestrator)

$\sigma \preceq_k \tau$ if and only if $f : \sigma \preceq_k \tau$ for some k-orchestrator f

Weak k-subcontract, formally

Set-theoretic interpretation of contracts

$$[\![\sigma]\!]^{\mathsf{w}}_k \stackrel{\text{def}}{=} \{\rho \mid \exists f : f : \rho \dashv\!\!\mid_k \sigma\}$$

Weak subcontract

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} [\![\sigma]\!]^{\mathsf{s}} \subseteq [\![\tau]\!]^{\mathsf{w}}_k$$

Proposition (existence of universal orchestrator)

$\sigma \preceq_k \tau$ if and only if $f : \sigma \preceq_k \tau$ for some k-orchestrator f

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \varepsilon\rangle.\langle b, \overline{b}\rangle$	b	$a.b$
$\langle \varepsilon, \overline{a}\rangle.\langle b, \overline{b}\rangle$	$a.b$	b
$\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle.\langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$	$b.a$	$a.b$

Theorem

$f : \rho \dashv\!|_k \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq_k \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \varepsilon \rangle . \langle b, \overline{b} \rangle$	b	$a.b$
$\langle \varepsilon, \overline{a} \rangle . \langle b, \overline{b} \rangle$	$a.b$	b
$\langle a, \varepsilon \rangle . \langle b, \varepsilon \rangle . \langle \varepsilon, \overline{b} \rangle . \langle \varepsilon, \overline{a} \rangle$	$b.a$	$a.b$

Theorem

$f : \rho \dashv\!\mid_k \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq_k \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \varepsilon\rangle.\langle b, \overline{b}\rangle$	b	$a.b$
$\langle \varepsilon, \overline{a}\rangle.\langle b, \overline{b}\rangle$	$a.b$	b
$\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle.\langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$	$b.a$	$a.b$

Theorem

$f : \rho \dashv|_k \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq_k \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \varepsilon\rangle.\langle b, \overline{b}\rangle$	b	$a.b$
$\langle \varepsilon, \overline{a}\rangle.\langle b, \overline{b}\rangle$	$a.b$	b
$\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle.\langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$	$b.a$	$a.b$

Theorem

$f : \rho \dashv\!|_k \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq_k \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrators as morphisms

Orchestrator application $f(\sigma)$

f	σ	$f(\sigma)$
$\langle a, \varepsilon\rangle.\langle b, \overline{b}\rangle$	b	$a.b$
$\langle \varepsilon, \overline{a}\rangle.\langle b, \overline{b}\rangle$	$a.b$	b
$\langle a, \varepsilon\rangle.\langle b, \varepsilon\rangle.\langle \varepsilon, \overline{b}\rangle.\langle \varepsilon, \overline{a}\rangle$	$b.a$	$a.b$

Theorem

$f : \rho \dashv\!\!|_k \sigma$ *if and only if* $\rho \dashv f(\sigma)$

Corollary

$f : \sigma \preceq_k \tau$ *if and only if* $\sigma \sqsubseteq f(\tau)$

Orchestrator composition and transitivity of $\preceq_k$

We've got a problem

$$\begin{aligned}
\sigma &\stackrel{\text{def}}{=} \overline{b} + \overline{d} \\
f &\stackrel{\text{def}}{=} \langle a, \varepsilon\rangle.\langle c, \varepsilon\rangle.(\langle \varepsilon, \overline{a}\rangle.\langle \overline{b}, b\rangle \vee \langle \varepsilon, \overline{c}\rangle.\langle \overline{d}, d\rangle) \\
g &\stackrel{\text{def}}{=} \langle a, \varepsilon\rangle.\langle \overline{b}, b\rangle \vee \langle c, \varepsilon\rangle.\langle \overline{d}, d\rangle
\end{aligned}$$

$$f(g(\sigma)) \simeq f(a.\overline{b} + c.\overline{d}) \simeq a.c.(\overline{b} \oplus \overline{d})$$

No single orchestrator can turn $\overline{b} + \overline{d}$ into $a.c.(\overline{b} \oplus \overline{d})$

Idea

Find an orchestrator $f \cdot g$ such that $f(g(\sigma)) \sqsubseteq (f \cdot g)(\sigma)$

Orchestrator composition and transitivity of $\preceq_k$

We've got a problem

$$\begin{aligned}
\sigma &\stackrel{\text{def}}{=} \overline{b} + \overline{d} \\
f &\stackrel{\text{def}}{=} \langle a, \varepsilon\rangle.\langle c, \varepsilon\rangle.(\langle \varepsilon, \overline{a}\rangle.\langle \overline{b}, b\rangle \vee \langle \varepsilon, \overline{c}\rangle.\langle \overline{d}, d\rangle) \\
g &\stackrel{\text{def}}{=} \langle a, \varepsilon\rangle.\langle \overline{b}, b\rangle \vee \langle c, \varepsilon\rangle.\langle \overline{d}, d\rangle
\end{aligned}$$

$$f(g(\sigma)) \simeq f(a.\overline{b} + c.\overline{d}) \simeq a.c.(\overline{b} \oplus \overline{d})$$

No single orchestrator can turn $\overline{b} + \overline{d}$ into $a.c.(\overline{b} \oplus \overline{d})$

Idea

Find an orchestrator $f \cdot g$ such that $f(g(\sigma)) \sqsubseteq (f \cdot g)(\sigma)$

Orchestrator composition and transitivity of $\preceq_k$

We've got a problem

$$\begin{aligned}
\sigma &\stackrel{\text{def}}{=} \overline{b} + \overline{d} \\
f &\stackrel{\text{def}}{=} \langle a, \varepsilon\rangle.\langle c, \varepsilon\rangle.(\langle \varepsilon, \overline{a}\rangle.\langle \overline{b}, b\rangle \vee \langle \varepsilon, \overline{c}\rangle.\langle \overline{d}, d\rangle) \\
g &\stackrel{\text{def}}{=} \langle a, \varepsilon\rangle.\langle \overline{b}, b\rangle \vee \langle c, \varepsilon\rangle.\langle \overline{d}, d\rangle
\end{aligned}$$

$$f(g(\sigma)) \simeq f(a.\overline{b} + c.\overline{d}) \simeq a.c.(\overline{b} \oplus \overline{d})$$

No single orchestrator can turn $\overline{b} + \overline{d}$ into $a.c.(\overline{b} \oplus \overline{d})$

Idea

Find an orchestrator $f \cdot g$ such that $f(g(\sigma)) \sqsubseteq (f \cdot g)(\sigma)$

Orchestrator composition and transitivity of $\preceq_k$

We've got a problem

$$\begin{aligned}
\sigma &\stackrel{\text{def}}{=} \overline{b} + \overline{d} \\
f &\stackrel{\text{def}}{=} \langle a, \varepsilon\rangle.\langle c, \varepsilon\rangle.(\langle \varepsilon, \overline{a}\rangle.\langle \overline{b}, b\rangle \vee \langle \varepsilon, \overline{c}\rangle.\langle \overline{d}, d\rangle) \\
g &\stackrel{\text{def}}{=} \langle a, \varepsilon\rangle.\langle \overline{b}, b\rangle \vee \langle c, \varepsilon\rangle.\langle \overline{d}, d\rangle
\end{aligned}$$

$$f(g(\sigma)) \simeq f(a.\overline{b} + c.\overline{d}) \simeq a.c.(\overline{b} \oplus \overline{d})$$

No single orchestrator can turn $\overline{b} + \overline{d}$ into $a.c.(\overline{b} \oplus \overline{d})$

Idea

Find an orchestrator $f \cdot g$ such that $f(g(\sigma)) \sqsubseteq (f \cdot g)(\sigma)$

Orchestrator composition, formally

$$f \cdot g \stackrel{\text{def}}{=} \bigvee_{f \xrightarrow{\langle \alpha, \varepsilon \rangle} f'} \langle \alpha, \varepsilon \rangle.(f' \cdot g) \vee \bigvee_{g \xrightarrow{\langle \varepsilon, \overline{\alpha} \rangle} g'} \langle \varepsilon, \overline{\alpha} \rangle.(f \cdot g')$$
$$\vee \bigvee_{f \xrightarrow{\langle \varphi, \overline{\alpha} \rangle} f', g \xrightarrow{\langle \alpha, \varphi' \rangle} g', \varphi\varphi' \neq \varepsilon} \langle \varphi, \varphi' \rangle.(f' \cdot g')$$
$$\vee \bigvee_{f \xrightarrow{\langle \varepsilon, \overline{\alpha} \rangle} f', g \xrightarrow{\langle \alpha, \varepsilon \rangle} g'} (f' \cdot g')$$

Theorem

$f(g(\sigma)) \sqsubseteq (f \cdot g)(\sigma)$

Deduction system for $\preceq_k$

No complete deduction system for $\preceq_k$ is known

(swap-inputs)	(swap-outputs)	(postpone-input)
$a.b.\sigma = b.a.\sigma$	$\overline{a}.\overline{b}.\sigma = \overline{b}.\overline{a}.\sigma$	$a.\overline{b}.\sigma \leq \overline{b}.a.\sigma$

Deduction system for $\preceq_k$

No complete deduction system for $\preceq_k$ is known

(swap-inputs)	(swap-outputs)	(postpone-input)
$a.b.\sigma = b.a.\sigma$	$\overline{a}.\overline{b}.\sigma = \overline{b}.\overline{a}.\sigma$	$a.\overline{b}.\sigma \leq \overline{b}.a.\sigma$

Duality

Duality

$$\rho^{\perp}$$

the $\preceq$-smallest service that satisfies ρ

Duality, examples

ρ	$\rho^{\perp}$
$a.\mathsf{e}$	
$a.\mathsf{e} \oplus b.\mathsf{e}$	
$a.\mathsf{e} + b.\mathsf{e}$	
e	
$a.\overline{b}.\mathsf{e} + a.\overline{c}.\mathsf{e}$	
0	
$a.\mathsf{e} \oplus 0$	
$a.0$	
$\dots$	

Definition (viable contract)

ρ is *viable* if there exists σ such that $\rho \dashv \sigma$

Duality, examples

ρ	$\rho^{\perp}$
$a.\mathsf{e}$	$\overline{a}$
$a.\mathsf{e} \oplus b.\mathsf{e}$	
$a.\mathsf{e} + b.\mathsf{e}$	
e	
$a.\overline{b}.\mathsf{e} + a.\overline{c}.\mathsf{e}$	
0	
$a.\mathsf{e} \oplus 0$	
$a.0$	
...	

Definition (viable contract)

ρ is *viable* if there exists σ such that $\rho \dashv \sigma$

Duality, examples

ρ	$\rho^{\perp}$
$a.\mathsf{e}$	$\overline{a}$
$a.\mathsf{e} \oplus b.\mathsf{e}$	$\overline{a} + \overline{b}$
$a.\mathsf{e} + b.\mathsf{e}$	
e	
$a.\overline{b}.\mathsf{e} + a.\overline{c}.\mathsf{e}$	
0	
$a.\mathsf{e} \oplus 0$	
$a.0$	
$\ldots$	

Definition (viable contract)

ρ is *viable* if there exists σ such that $\rho \dashv \sigma$

Duality, examples

ρ	$\rho^\perp$
$a.\mathsf{e}$	$\overline{a}$
$a.\mathsf{e} \oplus b.\mathsf{e}$	$\overline{a} + \overline{b}$
$a.\mathsf{e} + b.\mathsf{e}$	$\overline{a} \oplus \overline{b}$
e	
$a.\overline{b}.\mathsf{e} + a.\overline{c}.\mathsf{e}$	
0	
$a.\mathsf{e} \oplus 0$	
$a.0$	
$\dots$	

Definition (viable contract)

ρ is *viable* if there exists σ such that $\rho \dashv \sigma$

Duality, examples

ρ	$\rho^{\perp}$
$a.\mathsf{e}$	$\overline{a}$
$a.\mathsf{e} \oplus b.\mathsf{e}$	$\overline{a} + \overline{b}$
$a.\mathsf{e} + b.\mathsf{e}$	$\overline{a} \oplus \overline{b}$
e	0
$a.\overline{b}.\mathsf{e} + a.\overline{c}.\mathsf{e}$	
0	
$a.\mathsf{e} \oplus 0$	
$a.0$	
$\dots$	

Definition (viable contract)

ρ is *viable* if there exists σ such that $\rho \dashv \sigma$

Duality, examples

ρ	$\rho^\perp$
$a.\mathsf{e}$	$\overline{a}$
$a.\mathsf{e} \oplus b.\mathsf{e}$	$\overline{a} + \overline{b}$
$a.\mathsf{e} + b.\mathsf{e}$	$\overline{a} \oplus \overline{b}$
e	0
$a.\overline{b}.\mathsf{e} + a.\overline{c}.\mathsf{e}$	$\overline{a}.(b + c)$
0	
$a.\mathsf{e} \oplus 0$	
$a.0$	
$\dots$	

Definition (viable contract)

ρ is *viable* if there exists σ such that $\rho \dashv \sigma$

Duality, examples

ρ	$\rho^{\perp}$
$a.\mathsf{e}$	$\overline{a}$
$a.\mathsf{e} \oplus b.\mathsf{e}$	$\overline{a} + \overline{b}$
$a.\mathsf{e} + b.\mathsf{e}$	$\overline{a} \oplus \overline{b}$
e	0
$a.\overline{b}.\mathsf{e} + a.\overline{c}.\mathsf{e}$	$\overline{a}.(b + c)$
0	☹
$a.\mathsf{e} \oplus 0$	☹
$a.0$	☹
$\dots$	

Definition (viable contract)

ρ is *viable* if there exists σ such that $\rho \dashv \sigma$

Duality, formally

$$\rho^{\perp} \stackrel{\text{def}}{=} \sum_{\rho \Downarrow \mathsf{r}, \mathsf{e} \notin \mathsf{r}} \bigoplus_{\alpha \in \mathsf{r}, \mathsf{viable}(\rho(\alpha))} \overline{\alpha}.\rho(\alpha)^{\perp}$$

Theorem

Let ρ be a viable client contract. Then

1. $\rho \dashv \rho^{\perp}$
2. $\rho \dashv \sigma$ *implies* $\rho^{\perp} \preceq_0 \sigma$

Recursive behaviors

Finite syntax for finite behaviors

σ	$::=$		**contract**
		0	(null)
	$\mid$	$\alpha.\sigma$	(action prefix)
	$\mid$	$\sigma + \sigma$	(external choice)
	$\mid$	$\sigma \oplus \sigma$	(internal choice)

α	$::=$		**action**
		a	(input)
	$\mid$	$\overline{a}$	(output)

What about recursive behavior?

Describing recursive behavior

Many different ways. . .

$$\text{rec } X.a.X \oplus b.0$$

$$X = a.X \oplus b.0$$

$$\sigma^*$$

. . . **but it's just syntax!**

Infinite syntax for infinite behaviors

σ ::=		**contract**	
	0	(null)	
$\mid$	$\alpha.\sigma$	(action prefix)	
$\mid$	$\sigma + \sigma$	(external choice)	
$\mid$	$\sigma \oplus \sigma$	(internal choice)	

α ::=		**action**
	a	(input)
$\mid$	$\overline{a}$	(output)

Definition

The set of contracts is the set of *possibly infinite trees* generated by the grammar above such that

1. they have *finitely many different* subtrees
2. every *infinite branch* has *infinitely many prefixes*

Every *finite* contracts satisfies these conditions

Examples

$$\begin{aligned} X &= a.X \\ &= a.a.X \\ &= \cdots \end{aligned}$$ ☺

$$\begin{aligned} X &= a.X \oplus b.0 \\ &= a.(a.X \oplus b.0) \oplus b.0 \\ &= a.(a.(a.X \oplus b.0) \oplus b.0) \oplus b.0 \\ &= a.(a.(a.(a.X \oplus b.0) \oplus b.0) \oplus b.0) \oplus b.0 \\ &= \cdots \end{aligned}$$ ☺

$$X = X + X$$ ☹

$$X = X \oplus X$$ ☹

$$X = X$$ ☹

Recursion: summary

- all the results stated previously still hold (coinduction)
- use your own preferred syntax (but beware of Kleene $*$)

Why does it work?

Proposition

Let $D(\sigma) \stackrel{\text{def}}{=} \{\sigma' \mid \sigma \stackrel{\varphi}{\Longrightarrow} \sigma'\}$. *Then* $D(\sigma)$ *is finite for every* σ

Algorithm

Towards an algorithm for deciding $\preceq_k$

Problem: the orchestrator is not necessarily unique

$$
\begin{array}{rcrcl}
0 & : & a \oplus b \oplus 0 & \preceq & a + b \\
\langle a, \overline{a} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b \\
\langle b, \overline{b} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b \\
\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b
\end{array}
$$

$f \leqslant g$: g is better (more permissive) than f

$$f \leqslant g \overset{\text{def}}{\iff} [\![f]\!] \subseteq [\![g]\!]$$

Idea

- synthesize the best orchestrator
- the best orchestrator is unique

Towards an algorithm for deciding $\preceq_k$

Problem: the orchestrator is not necessarily unique

$$\begin{array}{rcccc} 0 & : & a \oplus b \oplus 0 & \preceq & a + b \\ \langle a, \overline{a} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b \\ \langle b, \overline{b} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b \\ \langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b \end{array}$$

$f \leqslant g$: g is better (more permissive) than f

$$f \leqslant g \stackrel{\text{def}}{\iff} [\![f]\!] \subseteq [\![g]\!]$$

Idea

- synthesize the best orchestrator
- the best orchestrator is unique

Towards an algorithm for deciding $\preceq_k$

Problem: the orchestrator is not necessarily unique

$$
\begin{array}{rcrcl}
0 & : & a \oplus b \oplus 0 & \preceq & a + b \\
\langle a, \overline{a} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b \\
\langle b, \overline{b} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b \\
\langle a, \overline{a} \rangle \vee \langle b, \overline{b} \rangle & : & a \oplus b \oplus 0 & \preceq & a + b
\end{array}
$$

$f \leqslant g$: g is better (more permissive) than f

$$
f \leqslant g \stackrel{\text{def}}{\iff} [\![f]\!] \subseteq [\![g]\!]
$$

Idea

- synthesize the best orchestrator
- the best orchestrator is unique

Deciding $\preceq_k$

$$\frac{\begin{array}{c} \mathsf{a}_r = \{\langle \varphi, \overline{\varphi}' \rangle \mid \sigma \stackrel{\varphi}{\Longrightarrow}, \tau \stackrel{\varphi'}{\Longrightarrow}, \mathbb{B} \vdash_k \langle \varphi, \overline{\varphi}' \rangle\} \\ \mathsf{a} = \{\langle \varphi, \overline{\varphi}' \rangle \in \mathsf{a}_r \mid \mathbb{B}\langle \varphi, \overline{\varphi}' \rangle \vdash_k f_{\langle \varphi, \overline{\varphi}' \rangle} : \sigma(\varphi) \trianglelefteq \tau(\varphi')\} \\ \tau \Downarrow \mathsf{s} \Rightarrow (\exists \mathsf{r} : \sigma \Downarrow \mathsf{r} \wedge \mathsf{r} \subseteq \mathsf{a} \circ \mathsf{s}) \vee (\emptyset \bullet \mathsf{a}) \cap \overline{\mathsf{s}} \neq \emptyset \end{array}}{\mathbb{B} \vdash_k \bigvee_{\mu \in \mathsf{a}} \mu.f_\mu : \sigma \trianglelefteq \tau}$$

Theorem

The following properties hold:

1. *(termination) the algorithm always terminates*
2. *(correctness)* $f : \sigma \trianglelefteq_k \tau$ *implies that* f *has rank* k *and* $f : \sigma \preceq_k \tau$
3. *(completeness)* $f : \sigma \preceq_k \tau$ *implies* $g : \sigma \trianglelefteq_k \tau$ *for some* $g \geqslant f$

An example from Wil's lecture (1/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_1 \stackrel{\text{def}}{=} \overline{\mathit{order}}.\mathit{food}.\overline{\mathit{money}}.\mathsf{e}$$

$$\rho_1^\perp = \mathit{order}.\overline{\mathit{food}}.\mathit{money}$$

$$f_1 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \mathit{money}, \overline{\mathit{money}} \rangle$$

An example from Wil's lecture (1/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_1 \stackrel{\text{def}}{=} \overline{\mathit{order}}.\mathit{food}.\overline{\mathit{money}}.\mathsf{e}$$

$$\rho_1^\perp = \mathit{order}.\overline{\mathit{food}}.\mathit{money}$$

$$f_1 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \mathit{money}, \overline{\mathit{money}} \rangle$$

An example from Wil's lecture (1/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_1 \stackrel{\text{def}}{=} \overline{\mathit{order}}.\mathit{food}.\overline{\mathit{money}}.\mathsf{e}$$

$$\rho_1^\perp = \mathit{order}.\overline{\mathit{food}}.\mathit{money}$$

$$f_1 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \mathit{money}, \overline{\mathit{money}} \rangle$$

An example from Wil's lecture (2/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_2 \stackrel{\text{def}}{=} \overline{\mathit{order}}.(\mathit{food}.\overline{\mathit{money}}.\mathsf{e} + \overline{\mathit{money}}.\mathit{food}.\mathsf{e})$$

$$\rho_2^{\perp} = \mathit{order}.(\overline{\mathit{food}}.\mathit{money} \oplus \mathit{money}.\overline{\mathit{food}})$$

$$f_2 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \mathit{money}, \overline{\mathit{money}} \rangle$$

An example from Wil's lecture (2/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_2 \stackrel{\text{def}}{=} \overline{\mathit{order}}.(\mathit{food}.\overline{\mathit{money}}.\mathsf{e} + \overline{\mathit{money}}.\mathit{food}.\mathsf{e})$$

$$\rho_2^\perp = \mathit{order}.(\overline{\mathit{food}}.\mathit{money} \oplus \mathit{money}.\overline{\mathit{food}})$$

$$f_2 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \mathit{money}, \overline{\mathit{money}} \rangle$$

An example from Wil's lecture (2/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_2 \stackrel{\text{def}}{=} \overline{\mathit{order}}.(\mathit{food}.\overline{\mathit{money}}.\mathsf{e} + \overline{\mathit{money}}.\mathit{food}.\mathsf{e})$$

$$\rho_2^\perp = \mathit{order}.(\overline{\mathit{food}}.\mathit{money} \oplus \mathit{money}.\overline{\mathit{food}})$$

$$f_2 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \mathit{money}, \overline{\mathit{money}} \rangle$$

An example from Wil's lecture (3/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_3 \stackrel{\text{def}}{=} \overline{\mathit{order}}.\overline{\mathit{money}}.\mathit{food}.\mathsf{e}$$

$$\rho_3^{\perp} = \mathit{order}.\mathit{money}.\overline{\mathit{food}}$$

$$f_3 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \mathit{money}, \varepsilon \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \varepsilon, \overline{\mathit{money}} \rangle$$

An example from Wil's lecture (3/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_3 \stackrel{\text{def}}{=} \overline{\mathit{order}}.\overline{\mathit{money}}.\mathit{food}.\mathsf{e}$$

$$\rho_3^\perp = \mathit{order}.\mathit{money}.\overline{\mathit{food}}$$

$$f_3 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \mathit{money}, \varepsilon \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \varepsilon, \overline{\mathit{money}} \rangle$$

An example from Wil's lecture (3/3)

$$\sigma \stackrel{\text{def}}{=} \mathit{order}.(\mathit{money} + \overline{\mathit{food}}.\mathit{money})$$

$$\rho_3 \stackrel{\text{def}}{=} \overline{\mathit{order}}.\overline{\mathit{money}}.\mathit{food}.\mathsf{e}$$

$$\rho_3^\perp = \mathit{order}.\mathit{money}.\overline{\mathit{food}}$$

$$f_3 = \langle \mathit{order}, \overline{\mathit{order}} \rangle.\langle \mathit{money}, \varepsilon \rangle.\langle \overline{\mathit{food}}, \mathit{food} \rangle.\langle \varepsilon, \overline{\mathit{money}} \rangle$$

Conclusions

Wrap-up

Subcontract relation

- tool for *searching* and *reasoning about* services by their contracts (= behavioral types)
- $\preceq$ combines reduction, extension, and permutation into a single preorder
- $\preceq$ gives safe substitution of services modulo orchestration
- $\preceq$ is decidable

(Simple) orchestrators

- have nice properties (universality, compositionality)
- can be automatically synthesized

Contracts vs session types

What is being typed

- contract = type of a process
- session type = type of a channel
- session type = type of a process projected on a channel

Testing approach

- session types: subtyping *preserves* correctness
- contracts: compliance *defines* subcontracting

Contracts vs session types

What is being typed

- contract = type of a process
- session type = type of a channel
- session type = type of a process projected on a channel

Testing approach

- session types: subtyping *preserves* correctness
- contracts: compliance *defines* subcontracting

Contracts vs session types

What is being typed

- contract = type of a process
- session type = type of a channel
- session type = type of a process projected on a channel

Testing approach

- session types: subtyping *preserves* correctness
- contracts: compliance *defines* subcontracting

Contracts vs session types

What is being typed

- contract = type of a process
- session type = type of a channel
- session type = type of a process projected on a channel

Testing approach

- session types: subtyping *preserves* correctness
- contracts: compliance *defines* subcontracting

Essential bibliography

Contracts with static interfaces and divergence

- C. Laneve, L. Padovani. **The *must* preorder revisited** (CONCUR'07)

Synthesis of orchestrators

- G. Castagna, N. Gesbert, and L. Padovani. **A theory of contracts for Web services** (POPL'08)
- G. Castagna, N. Gesbert, and L. Padovani. **A theory of contracts for Web Services**. (ACM TOPLAS 2009) *to appear*
- L. Padovani. **Contract-directed synthesis of simple orchestrators** (CONCUR'08)

Describing name-passing in contracts

- G. Castagna, L. Padovani. **Contracts for Mobile Processes** (CONCUR'09)