

Contract-based Discovery and Adaptation of Web Services

Luca Padovani

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

June 26, 2010

Contracts for Web services

Technologies

- behavioral descriptions of Web services (wsdl, wscl, ws-bpel, ...)
- repositories of Web services descriptions (uddi)

Goals

- *search 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 one

We propose

- **contracts** = abstractions of Web services' behavior

Finding Web services by contract

Compliance = client's satisfaction

$$\rho \dashv \sigma$$

Running a query with *compliance*

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

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

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

The quest for \preceq

Desiderata

- **reduction** of nondeterminism
- **extension** of functionalities
- some **permutation** of messages

$$a \oplus b \preceq a$$

$$a \preceq a + b$$

$$a.c \preceq c.a$$

Facts

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

Solution

- use (simple) **orchestrators**

Outline

- ① Introduction
- ② Basic theory of contracts
- ③ Orchestrators
- ④ Buffered orchestrators
- ⑤ Conclusions

Outline

- 1 Introduction
- 2 Basic theory of contracts**
- 3 Orchestrators
- 4 Buffered orchestrators
- 5 Conclusions

A ws-bpel service

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

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

    <switch>
      <case condition="getVariableData(InStock) == 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
I InStock
C Charge
S Ship
R Refund

Traces

{OICSO, OCISO, OICRO, OCIRO, OICO, OCIO}

Branching points

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

Contracts

Syntax

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

Example

$$0.(\bar{I}.\bar{C}.I.C.(\bar{S}.\bar{0} \oplus \bar{R}.\bar{0} \oplus \bar{0})) + \bar{C}.\bar{I}.I.C.(\bar{S}.\bar{0} \oplus \bar{R}.\bar{0} \oplus \bar{0}))$$

Operational semantics

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

- R. De Nicola, M. Hennessy, **CCS without tau's**, 1987

Compliance, formally

Systems

$$\rho \parallel \sigma$$

System transitions

$$\frac{\rho \rightarrow \rho'}{\rho \parallel \sigma \rightarrow \rho' \parallel \sigma}$$

$$\frac{\sigma \rightarrow \sigma'}{\rho \parallel \sigma \rightarrow \rho \parallel \sigma'}$$

$$\frac{\rho \xrightarrow{\bar{\alpha}} \rho' \quad \sigma \xrightarrow{\alpha} \sigma'}{\rho \parallel \sigma \rightarrow \rho' \parallel \sigma'}$$

Compliance

$$\rho \dashv \sigma \stackrel{\text{def}}{\iff} \rho \parallel \sigma \implies \rho' \parallel \sigma' \dashv \text{ implies } \rho' \xrightarrow{e}$$

Compliance, examples

$$\bar{a}.e \oplus \bar{b}.e \dashv ? a + b \quad \text{😊}$$

$$\bar{a}.e \oplus \bar{b}.e \dashv ? a \oplus b \quad \text{😞}$$

$$e \dashv ? \sigma \quad \text{😊}$$

$$0 \dashv ? \sigma \quad \text{😞}$$

Subcontract, formally

Set-theoretic interpretation of contracts

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

Subcontract

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

$$\simeq = \sqsubseteq \cap \supseteq$$

Properties of strong subcontract

Internal choice = intersection

$$\llbracket \sigma \oplus \tau \rrbracket^s = \llbracket \sigma \rrbracket^s \cap \llbracket \tau \rrbracket^s$$

External choice \neq union

- there are clients in $\llbracket a + b \rrbracket^s$ that are not in $\llbracket a \rrbracket^s \cup \llbracket b \rrbracket^s$:

$$\bar{a}.e \oplus \bar{b}.e \in \llbracket a + b \rrbracket^s \quad \bar{a}.e \oplus \bar{b}.e \notin \llbracket a \rrbracket^s \cup \llbracket b \rrbracket^s$$

- sometimes $+$ is \oplus in disguise:

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

- interferences:

$$\bar{a}.e + \bar{b} \in \llbracket a \rrbracket^s \quad \bar{a}.e + \bar{b} \notin \llbracket a + b \rrbracket^s$$

Properties of strong subcontract

Proposition

\sqsubseteq is a precongurence

$$\sigma \sqsubseteq \tau \quad \Rightarrow \quad \left\{ \begin{array}{l} \alpha.\sigma \sqsubseteq \alpha.\tau \\ \sigma \oplus \sigma' \sqsubseteq \tau \oplus \sigma' \\ \sigma + \sigma' \sqsubseteq \tau + \sigma' \end{array} \right.$$

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

Deduction system for \sqsubseteq

$$(e1) \quad \sigma + \sigma = \sigma$$

$$(e2) \quad \sigma + \tau = \tau + \sigma$$

$$(e3) \quad \sigma + (\sigma' + \sigma'') = (\sigma + \sigma') + \sigma''$$

$$(e4) \quad \sigma + \mathbf{0} = \sigma$$

$$(i1) \quad \sigma \oplus \sigma = \sigma$$

$$(i2) \quad \sigma \oplus \tau = \tau \oplus \sigma$$

$$(i3) \quad \sigma \oplus (\sigma' \oplus \sigma'') = (\sigma \oplus \sigma') \oplus \sigma''$$

$$(d1) \quad \sigma + (\sigma' \oplus \sigma'') = (\sigma + \sigma') \oplus (\sigma + \sigma'')$$

$$(d2) \quad \sigma \oplus (\sigma' + \sigma'') = (\sigma \oplus \sigma') + (\sigma \oplus \sigma'')$$

$$(d3) \quad \alpha.\sigma + \alpha.\tau = \alpha.(\sigma \oplus \tau)$$

$$(d4) \quad \alpha.\sigma \oplus \alpha.\tau = \alpha.(\sigma \oplus \tau)$$

$$(red) \quad \sigma \oplus \tau \leq \sigma$$

Outline

- 1 Introduction
- 2 Basic theory of contracts
- 3 Orchestrators**
- 4 Buffered orchestrators
- 5 Conclusions


Limitations of \sqsubseteq

\sqsubseteq does not support extensions...

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

...because of **interferences**

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

$$\bar{a}.e + \bar{b} \not\sqsubseteq a + b$$


Not all deadlocks are equal

Deadlock due to client/service nondeterminism

$$\bar{a}.e \oplus \bar{b}.e \parallel a \rightarrow \bar{b}.e \parallel a$$

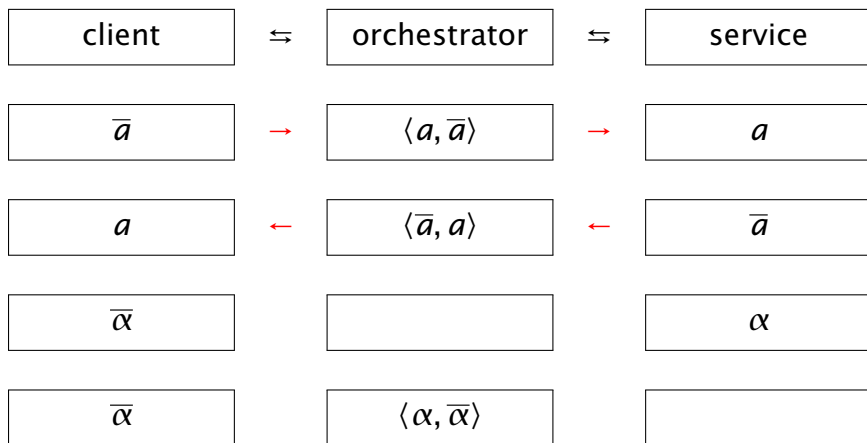
$$\bar{a}.e \parallel a \oplus b \rightarrow \bar{a}.e \parallel a$$

Deadlock due to **Murphy's law**

$$\bar{a}.e + \bar{b}.c.e \parallel a + b.\bar{d} \rightarrow e \parallel 0$$

$$\bar{a}.e + \bar{b}.c.e \parallel a + b.\bar{d} \rightarrow c.e \parallel \bar{d}$$

Idea: orchestrated interaction



Synchronous orchestrators

$f ::=$	orchestrator
0	(null)
$\mu.f$	(action prefix)
$f \vee f$	(union)

$\mu ::=$	orchestration action
$\langle a, \bar{a} \rangle$	(input/output)
$\langle \bar{a}, a \rangle$	(output/input)

Orchestrator semantics

Orchestrator transitions

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

Trace semantics for orchestrators

$$\llbracket f \rrbracket \stackrel{\text{def}}{=} \{ \mu_1 \cdots \mu_n \mid \exists g : f \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n} g \}$$

Orchestrated compliance, formally

Orchestrated systems

$$\rho \parallel_f \sigma$$

Orchestrated system transitions

$$\frac{\rho \rightarrow \rho'}{\rho \parallel_f \sigma \rightarrow \rho' \parallel_f \sigma} \quad \frac{\sigma \rightarrow \sigma'}{\rho \parallel_f \sigma \rightarrow \rho \parallel_f \sigma'}$$

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

Weak compliance

$$f : \rho \dashv\vdash \sigma \stackrel{\text{def}}{\iff} \rho \parallel_f \sigma \implies \rho' \parallel_{f'} \sigma' \dashv\vdash \text{implies } \rho' \xrightarrow{e}$$

Weak compliance, examples

$$\langle a, \bar{a} \rangle : e + a \dashv\vdash ? \bar{a} \quad \text{☹}$$

$$\mathbf{0} : e + a \dashv\vdash ? \bar{a} \quad \text{😊}$$

$$\langle a, \bar{a} \rangle \vee \langle b, \bar{b} \rangle : \bar{a}.e \oplus \bar{b}.e \dashv\vdash ? a + b \quad \text{😊}$$

$$\langle a, \bar{a} \rangle \vee \langle b, \bar{b} \rangle : \bar{a}.e \oplus \bar{b}.e \dashv\vdash ? a \oplus b \quad \text{☹}$$

Weak subcontract, formally

Set-theoretic interpretation of contracts

$$\llbracket \sigma \rrbracket^w \stackrel{\text{def}}{=} \{ \rho \mid \exists f : f : \rho \dashv\vdash \sigma \}$$

Weak subcontract

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} \llbracket \sigma \rrbracket^s \subseteq \llbracket \tau \rrbracket^w$$

Hmm...

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

Step 1: existence of universal orchestrator

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} \begin{array}{c} \forall \rho \dashv \sigma : \exists f : f : \rho \dashv \tau \\ \Updownarrow \\ \exists f : \forall \rho \dashv \sigma : f : \rho \dashv \tau \end{array}$$

Universal orchestrator

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

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

Step 2: relating \preceq and \sqsubseteq

Orchestrator application $f(\sigma)$

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

Theorem

- $f : \rho \dashv\vdash \sigma$ if and only if $\rho \dashv\vdash f(\sigma)$
- $f : \sigma \preceq \tau$ if and only if $\sigma \sqsubseteq f(\tau)$

Transitivity of \preceq (1/2)

$$f : \sigma \preceq \sigma' \quad g : \sigma' \preceq \tau \quad \stackrel{?}{\Rightarrow} \quad h : \sigma \preceq \tau$$

Proposition (Orchestrator application is monotone)

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

$$\sigma \sqsubseteq f(\sigma') \quad \sigma' \sqsubseteq g(\tau) \quad \Rightarrow \quad \sigma \sqsubseteq f(g(\tau))$$

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

Example

$$\begin{aligned} \langle a, \bar{a} \rangle \vee \langle b, \bar{b} \rangle . \langle c, \bar{c} \rangle & : a \oplus b.c \preceq a \\ \langle a, \bar{a} \rangle & : a \preceq a + b.d \\ \color{red}{???} & : a \oplus b.c \preceq a + b.d \end{aligned}$$

Transitivity of \preceq (2/2)

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

$$\llbracket f \wedge g \rrbracket \stackrel{\text{def}}{=} \llbracket f \rrbracket \cap \llbracket g \rrbracket$$

Proposition

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

Deduction system for \preceq (fragment)

$$\text{(red)} \quad I(\sigma) : \sigma \oplus \tau \leq \sigma$$

$$\text{(width)} \quad \frac{I(\sigma) \wedge I(\tau) = \mathbf{0}}{I(\sigma) : \sigma \leq \sigma + \tau}$$

$$\text{(depth)} \quad \mathbf{0} : \mathbf{0} \leq \sigma$$

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

An example by Wil van der Aalst (1/3)

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

$$\rho \stackrel{\text{def}}{=} \overline{\text{order}} . \text{food} . \overline{\text{money}} . e$$

$$\rho^\perp = \text{order} . \overline{\text{food}} . \text{money}$$

$$f = \langle \text{order}, \overline{\text{order}} \rangle . \langle \overline{\text{food}}, \text{food} \rangle . \langle \text{money}, \overline{\text{money}} \rangle$$

An example by Wil van der Aalst (2/3)

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

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

$$\rho^\perp = \text{order} . (\overline{\text{food}} . \text{money} \oplus \text{money} . \overline{\text{food}})$$

$$f = \langle \text{order}, \overline{\text{order}} \rangle . \langle \overline{\text{food}}, \text{food} \rangle . \langle \text{money}, \overline{\text{money}} \rangle$$

Interpretations of orchestrators

As mediators

$$\rho \parallel_f \sigma$$

As assumptions on the environment

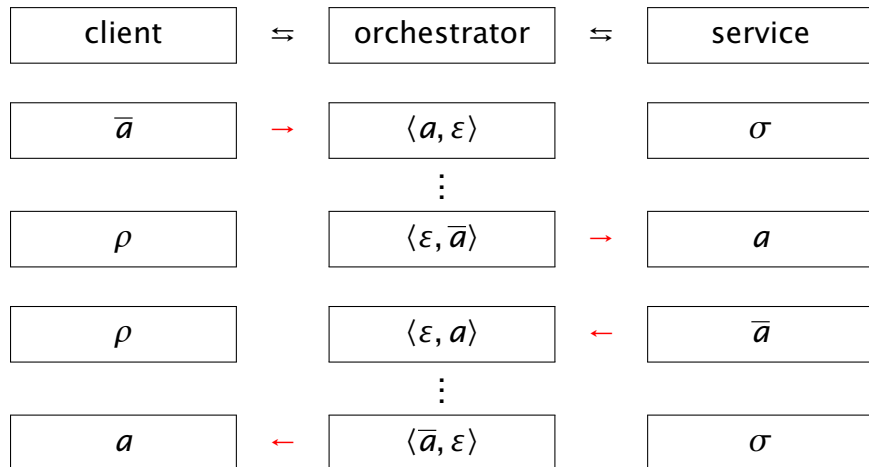
$$\langle a, \bar{a} \rangle : a \preceq a + b$$

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

Outline

- 1 Introduction
- 2 Basic theory of contracts
- 3 Orchestrators
- 4 Buffered orchestrators**
- 5 Conclusions

Buffered orchestrators



Syntax of buffered orchestrators

f	$::=$	orchestrator
	$\mathbf{0}$	(null)
	$ \mu.f$	(action prefix)
	$ f \vee f$	(union)

μ	$::=$	orchestration action
	$\langle \alpha, \bar{\alpha} \rangle$	(action/co-action)
	$ \langle \alpha, \varepsilon \rangle$	(action/-)
	$ \langle \varepsilon, \alpha \rangle$	(-/action)

Valid k -orchestrators

Not every orchestrator makes sense

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

Definition

Valid k -orchestrators are **directional**, **finite-state**, **fair**

Weak k -compliance, formally

Orchestrated systems

$$\rho \parallel_f \sigma$$

Orchestrated system transitions

...

$$\frac{\rho \xrightarrow{\bar{\alpha}} \rho' \quad f \xrightarrow{\langle \alpha, \varepsilon \rangle} f'}{\rho \parallel_f \sigma \rightarrow \rho' \parallel_{f'} \sigma'} \quad \frac{f \xrightarrow{\langle \varepsilon, \bar{\alpha} \rangle} f' \quad \sigma \xrightarrow{\alpha} \sigma'}{\rho \parallel_f \sigma \rightarrow \rho \parallel_{f'} \sigma'}$$

Weak k -compliance

$$f : \rho \dashv\!\!|_k \sigma \stackrel{\text{def}}{\iff} \rho \parallel_f \sigma \implies \rho' \parallel_{f'} \sigma' \dashv\!\!|_k \implies \rho' \xrightarrow{e}$$

Weak k -subcontract, formally

Set-theoretic interpretation of contracts

$$\llbracket \sigma \rrbracket_k^w \stackrel{\text{def}}{=} \{ \rho \mid \exists f : f : \rho \dashv\vdash_k \sigma \}$$

Weak subcontract

$$\sigma \preceq \tau \stackrel{\text{def}}{\iff} \llbracket \sigma \rrbracket^s \subseteq \llbracket \tau \rrbracket_k^w$$

Some properties of $\dashv\!\!|_k$ and \preceq_k

- existence of universal k -orchestrator (**harder!**)
- \preceq_k is transitive (**harder!**)
- no complete deduction system for \preceq_k is known

(swap-inputs)
 $a.b.\sigma = b.a.\sigma$

(swap-outputs)
 $\bar{a}.\bar{b}.\sigma = \bar{b}.\bar{a}.\sigma$

(postpone-input)
 $a.\bar{b}.\sigma \leq \bar{b}.a.\sigma$

An example by Wil van der Aalst (3/3)

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

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

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

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

Outline

- 1 Introduction
- 2 Basic theory of contracts
- 3 Orchestrators
- 4 Buffered orchestrators
- 5 Conclusions**

Wrap-up

Subcontract relation

- *searching for* and *reasoning about* services by their **contracts**
- \preceq combines **reduction**, **extension**, and **permutation** into a single preorder
- \preceq gives **safe substitution** of services modulo **orchestration**

(Simple) orchestrators

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

Other things you may want to know...

- how to compute σ^\perp
- how to deal with recursive (regular) behaviors
- how to *decide* \dashv , \preceq , and the like
- ...
- the symmetric theory

Essential bibliography

Orchestrators

- G. Castagna, N. Gesbert, and L. Padovani. **A theory of contracts for Web services** (POPL'08, ACM TOPLAS'09)
- L. Padovani. **Contract-directed synthesis of simple orchestrators** (CONCUR'08, TCS *to appear*)

Variations on the theme

Contracts and performances

- M. Bernardo, L. Padovani. **Performance-oriented comparison of Web services via client-specific testing preorders**, FMOODS'07

Contracts with static interfaces

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

Name-passing contracts

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