Testing equivalences revisited

ongoing research

Matthew Hennessy

(joint work with Giovanni Bernardi)

Koningswinter August 2013
Outline

Background

Web-service preorders

Axiomatisations

Future work
PhD theses by Italians

1984

- RDeN, MH: Testing Equivalences for Processes, ICALP 1983

2013

- Giovanni Bernardi: Behavioural Equivalences for Web Services. Trinity College Dublin
- GB, MH: Mutually Testing Processes, CONCUR 2013
PhD theses by Italians

1984

- RDeN, MH: Testing Equivalences for Processes, ICALP 1983

Processes understood by the tests they guarantee tests are other processes

2013

- Giovanni Bernardi: Behavioural Equivalences for Web Services. Trinity College Dublin
- GB, MH: Mutually Testing Processes, CONCUR 2013

Web-service contracts understood by compliance with other contracts
Processes testing processes

Process theory:
\[ p_1 \sqsubseteq p_2 \text{ if} \]
- Every test \( t \) guaranteed by \( p_1 \) is also guaranteed by \( p_2 \).
- A test \( t \) is a process with a success reporting mechanism

Web services:
Servers:
\[ s_1 \sqsubseteq_{svr} s_2 \text{ if every client satisfied by } s_1 \text{ is also satisfied by } s_2 \]

- What about client behaviour?
- What about peer behaviour? when all agents are both servers and clients
Processes testing processes

Process theory:
\[ p_1 \sqsubseteq p_2 \text{ if } \]
- Every test \( t \) guaranteed by \( p_1 \) is also guaranteed by \( p_2 \).
- A test \( t \) is a process with a *success* reporting mechanism

Web services:
Servers:
\[ s_1 \sqsubseteq_{svr} s_2 \text{ if every client satisfied by } s_1 \text{ is also satisfied by } s_2 \]

- What about *client* behaviour?
- What about *peer* behaviour?

when all agents are both *servers* and *clients*
Processes testing processes

Process theory:

\[ p_1 \sqsubseteq p_2 \text{ if } \]

- Every test \( t \) guaranteed by \( p_1 \) is also guaranteed by \( p_2 \).
- A test \( t \) is a process with a success reporting mechanism.

Web services:

Servers:

\[ s_1 \sqsubseteq_{svr} s_2 \text{ if every client satisfied by } s_1 \text{ is also satisfied by } s_2 \]

- What about client behaviour?
- What about peer behaviour? when all agents are both servers and clients
Web services via contracts

**Servers:**
\[ s_1 \preceq_{\text{svr}} s_2 \text{ if every client satisfied by } s_1 \text{ is also satisfied by } s_2 \]

*classical definition*

**Clients:**
\[ r_1 \preceq_{\text{clt}} r_2 \text{ if every server which satisfies } r_1 \text{ also satisfies } r_2 \]

*definition obvious*

**Peers:**
\[ p_1 \preceq_{\text{p2p}} p_2 \text{ if every peer which mutually satisfies } p_1 \text{ also mutually satisfies } p_2 \]

*various definitions possible*
Web-service contracts

- Contracts are states in an LTS with actions $\text{Act}_\tau$.

$$
\text{CCS}_1 : \quad p, q, r, s ::= 1 \mid A \mid \mu.p, \mu \in \text{Act}_\tau \mid \sum_{i \in I} p_i
$$

Note: $1 \xrightarrow{\checkmark}

Web-service interaction:

- Communication: $a \leftrightarrow \overline{a}$

\[\begin{align*}
q \xrightarrow{\lambda} q' \\
q \parallel p \xrightarrow{\lambda} q' \parallel p
\end{align*}\]

\[\begin{align*}
p \xrightarrow{\lambda} p' \\
q \parallel p \xrightarrow{\lambda} q \parallel p'
\end{align*}\]

\[\begin{align*}
q \xrightarrow{a} q' \\
p \xrightarrow{\overline{a}} p'
\end{align*}\]

\[\begin{align*}
q \parallel p \xrightarrow{\tau} q' \parallel p'
\end{align*}\]
Web-service contracts

- Contracts are states in an LTS with actions $\text{Act}_\tau$.

$$\text{CCS}_1 : \quad p, q, r, s ::= 1 \mid A \mid \mu. p, \mu \in \text{Act}_\tau \mid \sum_{i \in I} p_i$$

**Note:** $1 \xrightarrow{\checkmark}$

### Web-service interaction:

**Binary Communication:** $a \leftrightarrow \overline{a}$

**Standard Rules**

$$\begin{align*}
q \xrightarrow{\lambda} q' & \quad q \parallel p \xrightarrow{\lambda} q' \parallel p\\
q \parallel p \xrightarrow{\tau} q' \parallel p' & \quad \sum_{i \in I} p_i \xrightarrow{\lambda} p'\\
q \xrightarrow{a} q' & \quad p \xrightarrow{\overline{a}} p'\\
q \parallel p \xrightarrow{\tau} q' \parallel p' & \quad q \parallel p \xrightarrow{\lambda} q \parallel p'
\end{align*}$$
Web-service compliance preorders

Maximal interactions: $s_0 \parallel r_0 \xrightarrow{\tau} s_1 \parallel r_1 \xrightarrow{\tau} \ldots s_k \parallel r_k \ldots$

Client success: $r_k \xrightarrow{\checkmark}$ for some $k \geq 0$

Keeping clients happy:
- $s$ must $r$ if every maximal interaction from $s \parallel r$ is client-successful.

Server preorder:
- $s_1 \sqsubseteq_{svr} s_2$ if $s_1$ must $r$ implies $s_2$ must $r$ for every client $r$

Client preorder:
- $r_1 \sqsubseteq_{clt} r_2$ if $s$ must $r_1$ implies $s$ must $r_2$ for every server $s$
Web-service compliance preorders

Maximal interactions: $s_0 \parallel r_0 \xrightarrow{\tau} s_1 \parallel r_1 \xrightarrow{\tau} \ldots s_k \parallel r_k \ldots$

Client success: $r_k \checkmark$ for some $k \geq 0$

Keeping clients happy:

\textbf{s must } $r$ if every maximal interaction from $s \parallel r$ is client-successful.

Server preorder:

$s_1 \sqsubseteq_{svr} s_2$ if $s_1$ must $r$ implies $s_2$ must $r$

Client preorder:

$r_1 \sqsubseteq_{clt} r_2$ if $s$ must $r_1$ implies $s$ must $r_2$
Web-service compliance preorders

Maximal interactions: $s_0 \parallel r_0 \xrightarrow{\tau} s_1 \parallel r_1 \xrightarrow{\tau} \ldots s_k \parallel r_k \ldots$

Client success: $r_k \xrightarrow{\checkmark}$ for some $k \geq 0$

Keeping clients happy:
$s$ must $r$ if every maximal interaction from $s \parallel r$ is client-successful.

Server preorder:
$s_1 \sqsubseteq_{svr} s_2$ if $s_1$ must $r$ implies $s_2$ must $r$

Client preorder:
$r_1 \sqsubseteq_{clt} r_2$ if $s$ must $r_1$ implies $s$ must $r_2$
Clients

Clients are delicate: flowers

\[ a \cdot 1 \not\sqsubseteq_{\text{clt}} a \cdot \tau \cdot 1 \]

Try server \[ \bar{a}.(1 + \tau^\infty) \]

Well known equation \( \mu \cdot \tau.x = \mu \cdot x \) not valid

Clients are complicated:

\[ a.(b.0 + c.1) + a.(b.1 + c.0) \not\sqsubseteq_{\text{clt}} 0 \]
Clients

Clients are delicate: flowers

\[ a.1 \not\preceq_{\text{clt}} a.\tau.1 \]

Try server \[ \bar{a}.(1 + \tau^\infty) \]

Well known equation \[ \mu.\tau.x = \mu.x \] not valid

Clients are complicated:

\[ a.(b.0 + c.1) + a.(b.1 + c.0) \not\preceq_{\text{clt}} 0 \]
Clients

Clients are delicate: flowers

\[ a \cdot 1 \not\preceq_{\text{clt}} a \cdot \tau \cdot 1 \]

Try server \[ \overline{\bar{a}} \cdot (1 + \tau^\infty) \]

Well known equation \[ \mu \cdot \tau \cdot x = \mu \cdot x \] not valid

Clients are complicated:

\[ a \cdot (b \cdot 0 + c \cdot 1) + a \cdot (b \cdot 1 + c \cdot 0) \not\succeq_{\text{clt}} 0 \]
Clients

Clients are delicate: flowers

\[ a.1 \nleq_{\text{clt}} a.\tau.1 \]

Try server

\[ \overline{a}.(1 + \tau^\infty) \]

Well known equation \( \mu.\tau.x = \mu.x \) not valid

Clients are complicated:

\[ a.(b.0 + c.1) + a.(b.1 + c.0) \nleq_{\text{clt}} 0 \]
Peer-to-peer preorders

Maximal interactions: \[ p_0 \parallel q_0 \xrightarrow{\tau} p_1 \parallel q_1 \xrightarrow{\tau} \ldots p_k \parallel q_k \xrightarrow{\tau} \ldots \ldots \]

Peer-success:

\[ \checkmark p_k \rightarrow q_k \checkmark \text{ for some } k \geq 0 \]

\[ \checkmark \text{ for } i < k, p_i \nrightarrow \checkmark \text{ and } q_i \nrightarrow \checkmark \checkmark \]

\[ \text{Protocol: If I am happy so is my partner} \]

Happy peers:

\[ p \text{ must}^{p2p} q \text{ if every maximal interaction from } p \parallel q \text{ is peer-successful.} \]

Peer preorder:

\[ p_1 \sqsubseteq_{p2p} p_2 \text{ if } p_1 \text{ must}^{p2p} q \text{ implies } p_2 \text{ must}^{p2p} q \]

Variations:

Independent success repetitions allowed
Peer-to-peer preorders

Maximal interactions:

\[ p_0 \parallel q_0 \xrightarrow{\tau} p_1 \parallel q_1 \xrightarrow{\tau} \ldots p_k \parallel q_k \xrightarrow{\tau} \ldots \]

Peer-success:

- \( p_k \xrightarrow{\checkmark} q_k \xrightarrow{\checkmark} \) for some \( k \geq 0 \)
- for \( i < k \), \( p_i \not\xrightarrow{\checkmark} \) and \( q_i \not\xrightarrow{\checkmark} \)
- Protocol: If I am happy so is my partner

Happy peers:

- \( p \) must\(^{p2p} \) \( q \) if every maximal interaction from \( p \parallel q \) is peer-successful.

Peer preorder:

- \( p_1 \equiv_{p2p} p_2 \) if \( p_1 \) must\(^{p2p} \) \( q \) implies \( p_2 \) must\(^{p2p} \) \( q \)

Variations:

- Independent success repetitions allowed

see Concur 2013
Peer-to-peer preorders

Maximal interactions: \( p_0 \parallel q_0 \xrightarrow{\tau} p_1 \parallel q_1 \xrightarrow{\tau} \ldots p_k \parallel q_k \xrightarrow{\tau} \ldots \)

Peer-success:

1. \( p_k \xrightarrow{\checkmark} q_k \xrightarrow{\checkmark} \) for some \( k \geq 0 \)
2. For \( i < k \), \( p_i \not\xrightarrow{\checkmark} \) and \( q_i \not\xrightarrow{\checkmark} \)
3. Protocol: If I am happy so is my partner

Happy peers:

\( p \) must\(^{p2p} q \) if every maximal interaction from \( p \parallel q \) is peer-successful.

Peer preorder:

\( p_1 \preceq_{p2p} p_2 \) if \( p_1 \) must\(^{p2p} q \) implies \( p_2 \) must\(^{p2p} q \) for every peer \( q \)

Variations:

- Independent success repetitions allowed

see Concur 2013
Peer-to-peer preorders

Maximal interactions: \( p_0 \parallel q_0 \xrightarrow{\tau} p_1 \parallel q_1 \xrightarrow{\tau} \ldots p_k \parallel q_k \xrightarrow{\tau} \ldots \)

Peer-success:
1. \( p_k \xrightarrow{\checkmark} q_k \xrightarrow{\checkmark} \) for some \( k \geq 0 \)
2. for \( i < k \), \( p_i \not\xrightarrow{\checkmark} \) and \( q_i \not\xrightarrow{\checkmark} \)
3. Protocol: If I am happy so is my partner

Happy peers:
- \( p \) must\(^{p_{2p}} q \) if every maximal interaction from \( p \parallel q \) is peer-successful.

Peer preorder:
- \( p_1 \preceq_{p_{2p}} p_2 \) if \( p_1 \) must\(^{p_{2p}} q \) implies \( p_2 \) must\(^{p_{2p}} q \)

Variations:
- Independent success repetitions allowed

For more details, see Concur 2013
Peers

Peers are delicate: flowers

\[ c.a.(1 + b.1) \not\preceq_{p2p} c.(1 + a.(1 + b.1)) \]

Use partner peer \( \overline{c.a}.1 \)

Peers are complicated:

\[ \tau.p + \tau.1 \preceq_{p2p} 0 \]

regardless of \( p \)
Understanding behavioural preorders $p \sqsupseteq q$

- Use behavioural properties
  - traces, failures, acceptances

- Use denotational models

- Use algebraic equations/inequations
Understanding behavioural preorders $p \preceq q$

- Use behavioural properties
  - traces, failures, acceptances

  *Classical approach* see *Concur 2013 paper*

- Use denotational models
  *Not very popular not very easy*

- Use algebraic equations/inequations

  *Lets have a look*
Outline

Background

Web-service preorders

Axiomatisations

Future work
Axiomatisation:

Language $\text{CCS}_{1}^{f} : 1, 0, \Omega, \mu. - \mu \in \text{Act}_{\tau}, +$

Problem: well-known

$0 \not{\sim}_{\text{clt}} b. 0$ but $a. 1 + 0 \not{\sim}_{\text{clt}} a. 1 + b. 0$

$\not{\sim}_{*}$ is not preserved by $+$

Solution: standard

Use largest pre-congruence contained in $\not{\sim}_{*}$:

$p \not{\sim}_{*}^{c} q$ if $C[p] \not{\sim}_{*} C[q]$ for every context $C[~]$

Characterisation: standard

$p \not{\sim}_{*}^{c} q$ iff $f.1 + p \not{\sim}_{*} f.1 + q$

$\not{\sim}_{*}^{+}$

Notation: $p \not{\sim}_{*}^{+} q$
Axiomatisation:

Language $\text{CCS}_1^f : 1, 0, \Omega, \mu - \mu \in \text{Act}_\tau, +$

Problem: well-known

$0 \clt b.0$ but $a.1 + 0 \clt a.1 + b.0$

$\star$ is not preserved by $+$

Solution: standard

Use largest pre-congruence contained in $\star$:

$p \preceq^C q$ if $C[p] \subseteq_{\star} C[q]$ for every context $C[\ ]$

Characterisation: standard

$p \preceq^C q$ iff $f.1 + p \preceq_{\star} f.1 + q$

f fresh
Axiomatisation:

Language $\text{CCS}_1^f : 1, 0, \Omega, \mu - \mu \in \text{Act}_\tau$, +

Problem: well-known

$0 \sim \text{clt} b.0$ but $a.1 + 0 \sim \text{clt} a.1 + b.0$

$\sim_*$ is not preserved by $+$

Solution: standard

Use largest pre-congruence contained in $\sim_*$:

$p \sim_c q$ if $C[p] \sim C[q]$ for every context $C[]$

Characterisation: standard

$p \sim_c q$ iff $f.1 + p \sim f.1 + q$

Notation: $p \sim_+ q$
Axiomatising the server preorder 1983

For CCS\(^f\):

\((S1)\) \(\mu.x + \mu.y = \mu.(\tau.x + \tau.y)\)
\((S2)\) \(x + \tau.y = \tau.(x + y) + \tau.y\)
\((S3)\) \(\mu.x + \tau.(\mu.y + z) = \tau.(\mu.x + \mu.y + z)\)
\((S4)\) \(\tau.x + \tau.y \leq x\)
\((S5)\) \(\Omega \leq x\)

Problem: Clients/Peers are delicate
\((S1)\) \((S2)\) are not valid

Solution: two sorted equations

- \(x, y, \ldots\) variables only instantiated by \(p \xrightarrow{\ }\)
- \(x, y, \ldots\) arbitrary instantiation
Axiomatising the server preorder

For CCS\(^f\):

\begin{align*}
\text{(S1)} & \quad \mu.x + \mu.y = \mu.(\tau.x + \tau.y) \\
\text{(S2)} & \quad x + \tau.y = \tau.(x + y) + \tau.y \\
\text{(S3)} & \quad \mu.x + \tau.(\mu.y + z) = \tau.(\mu.x + \mu.y + z) \\
\text{(S4)} & \quad \tau.x + \tau.y \leq x \\
\text{(S5)} & \quad \Omega \leq x
\end{align*}

Problem: Clients/Peers are delicate

(S1) (S2) are not valid

Solution: two sorted equations

- \(x, y, \ldots\) variables only instantiated by \(p\)
- \(x, y, \ldots\) arbitrary instantiation
Axiomatising the server preorder \(1983\)

For CCS:\(f\):

\[
\begin{align*}
\text{(S1)} & \quad \mu.x + \mu.y = \mu.(\tau.x + \tau.y) \\
\text{(S2)} & \quad x + \tau.y = \tau.(x + y) + \tau.y \\
\text{(S3)} & \quad \mu.x + \tau.(\mu.y + z) = \tau.(\mu.x + \mu.y + z) \\
\text{(S4)} & \quad \tau.x + \tau.y \leq x \\
\text{(S5)} & \quad \Omega \leq x
\end{align*}
\]

Problem: Clients/Peers are delicate
\(\text{(S1)}\) \(\text{(S2)}\) are not valid

Solution: two sorted equations

\> \(x, y, \ldots\) variables only instantiated by \(p \xrightarrow{\checkmark}\)

\> \(x, y, \ldots\) arbitrary instantiation
Standard inequations

(S1a) \[ \mu.x + \mu.y = \mu.(\tau.x + \tau.y) \]
(S1b) \[ \tau.x + \tau.y = \tau.(\tau.x + \tau.y) \]
(S2) \[ x + \tau.y = \tau.(x + y) + \tau.y \]
(S3) \[ \mu.x + \tau.(\mu.y + z) = \tau.(\mu.x + \mu.y + z) \]
(S4) \[ \tau.x + \tau.y \leq x \]
(S5) \[ \Omega \leq x \]

Characterising server preorder \( \sqsubseteq_{svr}^{c} \) over \( CCS_f \):

Add \[ 1 = 0 \]

1 is useless for servers

What about clients and peers?
Standard inequations

(S1a) $\mu.x + \mu.y = \mu.(\tau.x + \tau.y)$
(S1b) $\tau.x + \tau.y = \tau.(\tau.x + \tau.y)$
(S2) $x + \tau.y = \tau.(x + y) + \tau.y$
(S3) $\mu.x + \tau.(\mu.y + z) = \tau.(\mu.x + \mu.y + z)$
(S4) $\tau.x + \tau.y \leq x$
(S5) $\Omega \leq x$

Characterising server preorder $\simeq_{svr}^c$ over $CCS_f^i$:

Add $1 = 0$

1 is useless for servers

What about clients and peers?
Clients: axioms for $\simeq_{\text{clt}}$

1 is a top element: $x \leq 1$

- $f.1 + r \simeq_{\text{clt}} f.1 + 1$ for every client $r$
- $f.1 + r \simeq_{\text{clt}} f.1 + 1$ for every client $r$
- 1 is also an annihilator: $1 + r \simeq_{\text{clt}} 1$

0 is not a bottom element: $0 \not\simeq_{\text{clt}} a.0$

- $\overline{f}.1 + a.0$ must $f.1 + 0$
- $\overline{f}.0 + a.0$ must $f.1 + a.0$

$\tau.0$ is a bottom element: $\tau.0 \leq x$

- $f.1 + \tau.0 \simeq_{\text{clt}} r$ for every client $r$
- reason: no server satisfies $f.1 + \tau.0$
Clients: axioms for $\cong_{\text{clt}}$

1 is a top element: $x \leq 1$

- $f.1 + r \cong_{\text{clt}} f.1 + 1$ for every client $r$
- $f.1 + r \cong_{\text{clt}} f.1 + 1$ for every client $r$
- 1 is also an annihilator: $1 + r \cong_{\text{clt}} 1$

0 is not a bottom element: $0 \not\cong_{\text{clt}} a.0$

- $\overline{f}.1 + a.0$ must $f.1 + 0$
- $\overline{f}.0 + a.0$ must $f.1 + a.0$

$\tau.0$ is a bottom element: $\tau.0 \leq x$

- $f.1 + \tau.0 \cong_{\text{clt}} r$ for every client $r$
- reason: no server satisfies $f.1 + \tau.0$
Clients: axioms for $\simeq_{\text{clt}}$

1 is a top element: $x \leq 1$

- $f.1 + r \simeq_{\text{clt}} f.1 + 1$ for every client $r$
- $f.1 + r \simeq_{\text{clt}} f.1 + 1$ for every client $r$
- 1 is also an annihilator: $1 + r \simeq_{\text{clt}} 1$

0 is not a bottom element: $0 \not\simeq_{\text{clt}} a.0$

- $\overline{f}.1 + a.0$ must $f.1 + 0$
- $\overline{f}.0 + a.0$ must $f.1 + a.0$

τ.0 is a bottom element: $\tau.0 \leq x$

- $f.1 + \tau.0 \simeq_{\text{clt}} r$ for every client $r$
- reason: no server satisfies $f.1 + \tau.0$
Clients: axioms for $\sim_{clt}$

1 is a top element: $x \leq 1$

- $f.1 + r \sim_{clt} f.1 + 1$ for every client $r$
- $f.1 + r \sim_{clt} f.1 + 1$ for every client $r$
- 1 is also an annihilator: $1 + r \sim_{clt} 1$

0 is not a bottom element: $0 \not\sim_{clt} a.0$

- $\overline{f}.1 + a.0$ must $f.1 + 0$
- $\overline{f}.0 + a.0$ must $f.1 + a.0$

$\tau.0$ is a bottom element: $\tau.0 \leq x$

- $f.1 + \tau.0 \sim_{clt} r$ for every client $r$
- reason: no server satisfies $f.1 + \tau.0$
Client axiomatisation over $\text{CCS}_1^f$

Add to the standard inequations:

\[
\begin{align*}
\tau. 0 & \leq x & \text{bottom} \\
x & \leq 1 & \text{top} \\
1 & \leq 1 + x & \text{annihilator} \\
a. 0 & \leq 0 & \text{adding $a. 0$ does not help clients} \\
0 & \leq \mu. 1 & \text{adding $a. 1$ helps clients}
\end{align*}
\]
Peers: axioms for $\sqsubseteq_{p2p}$

1 is not a top element: $a.1 \sqsubseteq_{p2p} 1$

- $a.1 \text{ must}^{p2p} a.1$
- $a.1 \not\text{must}^{p2p} 1$

$\tau.1$ is another bottom element:

- $\bar{f}.1 + \tau.1 \text{ must}^{p2p} p$ for every peer $p$
- reason: no peer is satisfied by $f.1 + \tau.1$
Peers: axioms for $\sqsubseteq^C_{p2p}$

1 is not a top element: $a.1 \sqsubseteq_{p2p} 1$

- $a.1 \text{ must}^{p2p} a.1$
- $a.1 \not\text{must}^{p2p} 1$

τ.1 is another bottom element:

- $f.1 + \tau.1 \text{ must}^{p2p} p$ for every peer $p$
- reason: no peer is satisfied by $f.1 + \tau.1$
## Clients vs Peers

<table>
<thead>
<tr>
<th>Clients</th>
<th>Peers</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau.0 \leq x )</td>
<td>replace with ( x + 1 \leq 1 )</td>
</tr>
<tr>
<td>( x \leq 1 )</td>
<td>( x + 1 \leq 1 )</td>
</tr>
<tr>
<td>( 1 \leq 1 + x )</td>
<td></td>
</tr>
<tr>
<td>( a.0 \leq 0 )</td>
<td></td>
</tr>
<tr>
<td>( 0 \leq \mu.1 )</td>
<td>replace with ( \tau.1 \leq x )</td>
</tr>
</tbody>
</table>

Peer variations can also be captured.
## Clients vs Peers

<table>
<thead>
<tr>
<th>Clients</th>
<th>Peers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau \cdot 0 \leq x$</td>
<td>$\tau \cdot 1 \leq x$</td>
</tr>
<tr>
<td>$x \leq 1$</td>
<td>$x + 1 \leq 1$</td>
</tr>
<tr>
<td>$1 \leq 1 + x$</td>
<td>replace with $x + 1 \leq 1$</td>
</tr>
<tr>
<td>$a \cdot 0 \leq 0$</td>
<td></td>
</tr>
<tr>
<td>$0 \leq \mu \cdot 1$</td>
<td>replace with $\tau \cdot 1 \leq x$</td>
</tr>
</tbody>
</table>

Peer variations can also be captured
Outline

Background

Web-service preorders

Axiomatisations

Future work
Multi-agent systems

Definitions definitions definitions . . .

\[ r_1 \sqsubseteq_{clt}^n r_1' \quad \text{if} \]

\[ s \text{ must}_{clt}^n r_1 \parallel r_2 \ldots \parallel r_n \quad \text{implies} \quad s \text{ must}_{clt}^n r_1' \parallel r_2 \ldots \parallel r_n \]

for all servers \( s \) and clients possible clients \( r_2, \ldots r_n \)

Should:

(a) all \( r_1 \ldots r_n \) should report success?
(b) at least one of one of \( r_1 \ldots r_n \) should report success?

Conjecture:

With (a): \( r_1 \sqsubseteq_{clt} r_1' \iff r_1 \sqsubseteq_{clt}^n r_1', \ n \geq 1 \)
Multi-agent systems

Definitions

\[ r_1 \sqsubseteq^n _{\text{clt}} r'_1 \quad \text{if} \]

\[
\begin{array}{c}
\text{s must}^{\text{clt}} n r_1 || r_2 \ldots || r_n
\end{array}
\]

implies

\[
\begin{array}{c}
\text{s must}^{\text{clt}} n r'_1 || r_2 \ldots || r_n
\end{array}
\]

for all servers \( s \) and clients possible clients \( r_2, \ldots r_n \)

Should:

(a) all \( r_1 \ldots r_n \) should report success?

(b) at least one of one of \( r_1 \ldots r_n \) should report success?

Conjecture:

With (a):

\[ r_1 \sqsubseteq^n _{\text{clt}} r'_1 \iff r_1 \sqsubseteq^n _{\text{clt}} r'_1, \quad n \geq 1 \]
Multi-agent systems

Definitions definitions definitions ...

\[ r_1 \sim_{clt}^{n} r'_1 \text{ if} \]

\[
\begin{array}{c}
\text{s must}^{clt}_{n} r_1 \ || \ r_2 \ldots \ || \ r_n
\end{array}
\implies
\begin{array}{c}
\text{s must}^{clt}_{n} r'_1 \ || \ r_2 \ldots \ || \ r_n
\end{array}
\]

for all servers \( s \) and clients possible clients \( r_2, \ldots r_n \)

Should:

(a) all \( r_1 \ldots r_n \) should report success?

(b) at least one of one of \( r_1 \ldots r_n \) should report success?

Conjecture:

With (a): \( r_1 \sim_{clt} r'_1 \iff r_1 \sim_{clt}^{n} r'_1, \ n \geq 1 \)
References

