FORMAL VERIFICATION OF WEB SERVICES COMPOSITION USING LINEAR LOGIC AND THE Π-CALCULUS

Petros Papapanagiotou and Jacques Fleuriot
University of Edinburgh
Web Services Composition

- Mechanically combine Web Services for complex tasks

- In this work:
  - Offline
  - Formal
  - Exceptions
  - Non-functional properties (Quality driven)
Web Services Composition using Theorem Proving

- Rigorous approach – fully verified
- No ad-hoc implementations
- Trust!
- HOL Light:
  - Well respected HOL theorem prover
  - Extensible, flexible, programmable
  - Trusted:
    - LCF approach
    - Small, trusted core of axioms
  - But: requires expertise
Real Estate Example

- Home purchasing (Zhang et al. 2005)
- 10 available services
- Request: A single composite service
Example: Available Services
Example: Requested Service

5

User Input

Buyer

Settlement or Exception

Home Directory

Estate Agent

Contract Service

Title Search

Mortgage Service

Home Insurance

Criminal Service

House Alert

Settlement

Home Insurance

Buyer
Proofs-as-processes

- Similar to “formulae-as-types” (Curry-Howard 1980)
- Introduced by Abramsky (1994)
- Bellin and Scott (1994)

- Classical Linear Logic and π-calculus

- Systematically (deeply) embedded in a theorem prover:
  - Formalised proof rules: can be trusted
  - Verification
π-calculus

- Milner (1989)

<table>
<thead>
<tr>
<th>P ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>x(y).P</td>
</tr>
<tr>
<td>\bar{x}y?.P</td>
</tr>
<tr>
<td>(\nu x) P</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>P + P</td>
</tr>
</tbody>
</table>
Classical Linear Logic

- Girard (1987)

<table>
<thead>
<tr>
<th></th>
<th>Disjunction</th>
<th>Conjunction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multiplicative</strong></td>
<td>( \otimes )</td>
<td>( \otimes )</td>
</tr>
<tr>
<td><strong>Additive</strong></td>
<td>( \oplus )</td>
<td>( &amp; )</td>
</tr>
<tr>
<td><strong>Negation</strong></td>
<td>( \bot )</td>
<td>( \bot )</td>
</tr>
</tbody>
</table>

*Red: input  Blue: output*
Proofs-as-processes

Γ ⊢ π
CLL inference rule

\[
\vdash x : A, \ y : A^\perp \\
\vdash F, \ G \\
\vdash \bar{w} : \Gamma, \ x : A \vdash \bar{u} : \Delta, \ y : B \\
\vdash \bar{w} : \Gamma, \ \bar{u} : \Delta, \ z : A \otimes B
\]

\[\otimes (F, G) \bar{w} \bar{u} z = \nu x y (\bar{z}(xy)(F_{\bar{w}x} || G_{\bar{u}y}))\]

\[\nu^{x,y}_z (F) \bar{w} z = z(xy)F_{\bar{w}xy}\]

\[\nu^{x,y}_z (P) \bar{w} z = \nu x(z(uv)\bar{u}(x)P_{\bar{w}x})\]

\[\nu^{y}_z (Q) \bar{w} y = \nu y(z(uv)\bar{v}(y)Q_{\bar{w}y})\]

\[\&^{x,y}_z (P, Q) \bar{w} z = \nu u v(\bar{z}(uv)[u(x)P_{\bar{w}x} + v(y)Q_{\bar{w}y}])\]

\[\text{Cut}^{z}(F, G) \bar{w} \bar{v} = \nu z(F_{\bar{w}[z/x]] || G_{\bar{v}[z/y]})\]
<table>
<thead>
<tr>
<th>Buffer</th>
<th>Parallel</th>
<th>Choice</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\vdash x:A, y:A^\perp]</td>
<td>[I xy = y(a)\overline{x}(a)]</td>
<td>[\bigotimes(F, G)\overline{w}u z = \nu x y(\overline{z}(xy)(F_{\overline{w}x}</td>
<td></td>
</tr>
<tr>
<td>[\vdash \overline{w}:\Gamma, x:A, y:B]</td>
<td>[\vdash \overline{w}:\Gamma, \overline{u}:\Delta, z:A \otimes B]</td>
<td>[\bigotimes(F, G)\overline{w}u z = \nu x y(\overline{z}(xy)(F_{\overline{w}x}</td>
<td></td>
</tr>
<tr>
<td>[\vdash \overline{w}:\Gamma, x:A]</td>
<td>[\vdash \overline{w}:\Gamma, y:B]</td>
<td>[\bigotimes(P, Q)\overline{w} z = \nu y(zuv)\overline{u}(y)[u(x)P_{\overline{w}x} + v(y)Q_{\overline{w}y}]]</td>
<td>[\xi(P, Q)\overline{w} z = \nu y(zuv)\overline{u}(y)[u(x)P_{\overline{w}x} + v(y)Q_{\overline{w}y}]]</td>
</tr>
<tr>
<td>[\vdash \overline{u}:\Gamma, x:C]</td>
<td>[\vdash \overline{u}:\Delta]</td>
<td>[\xi(P, Q)\overline{w} z = \nu y(zuv)\overline{u}(y)[u(x)P_{\overline{w}x} + v(y)Q_{\overline{w}y}]]</td>
<td>[\xi(P, Q)\overline{w} z = \nu y(zuv)\overline{u}(y)[u(x)P_{\overline{w}x} + v(y)Q_{\overline{w}y}]]</td>
</tr>
</tbody>
</table>
WS Composition using proofs-as-processes

Translate to CLL
Prove Requested Service
Extract $\pi$-calculus term
Realisation ...

...
WS Composition using proofs-as-processes

- Translate to CLL
- Prove Requested Service
- Extract $\pi$-calculus term
- Realisation...
Web Services in Classical Linear Logic

\[ \vdash \vec{P}^\perp, \vec{I}^\perp, ((\bigotimes_i \vec{F}) \otimes (\bigotimes_i \vec{O})) \oplus E \]

Where: \( \bigotimes_i (\alpha_1, \alpha_2, ..., \alpha_n) = \alpha_1 \otimes \alpha_2 \otimes ... \otimes \alpha_n \), \( i \geq 0 \)

\( \vec{P} \): Preconditions
\( \vec{F} \): Postconditions / Effects
\( \vec{I} \): Input
\( \vec{O} \): Output
\( \vec{E} \): Exception
Real Estate Example specified in CLL

1. HomeDir: ⊬ HOME_CRITERIA, HOME_LISTING
2. CriminalService: ⊬ REGION, CRIMINAL_ACT
3. HouseAlert: ⊬ HOME_LISTING, CRIMINAL_ACT, DESIRED_LEVEL, HOME_TITLE_ID \(\otimes\) HOME_AGENT_ID \(\otimes\) HOME_DESC
4. Buyer: ⊬ HOME_DESC, HOME_OFFER
5. EstateAgentSeller: ⊬ HOME_AGENT_ID, HOME_OFFER, ACCEPTED_OFFER \(\oplus\) REJECTED_OFFER
6. MortgageService: ⊬ CLIENT_INFO, PREAPPROVAL \(\oplus\) EXM
7. ContractService: ⊬ PREAPPROVAL, ACCEPTED_OFFER, CONTRACT
8. TitleSearch: ⊬ HOME_TITLE_ID, TITLE \(\otimes\) (HOME_INSURANCE \(\oplus\) HOME_INS_ID)
9. HomeInsurance: ⊬ HOME_INS_ID, HOME_INS
10. Settlement: ⊬ TITLE, CONTRACT, HOME_INS, SETTLEMENT
Real Estate Request in CLL

⊢ HOME_CRITERIA⊥, REGION⊥, DESIRED_LEVEL⊥, CLIENT_INFO⊥, SETTLEMENT ⊕ EXCEPTION
Real Estate Request in CLL

⊢ HOME_CRITERIA⊥, REGION⊥, DESIRED_LEVEL⊥, CLIENT_INFO⊥, SETTLEMENT ⊕ EXCEPTION

What is the final exception?

☐ TP: uninstantiated metavariables + unification
☐ Rely on trusted built-in procedures
WS Composition using proofs-as-processes

Translate to CLL

Prove Requested Service

Extract $\pi$-calculus term

Realisation ...

18
Example sub-proof

- **Multiple outputs:**
  
  HouseAlert: \( \vdash \text{HL}^\perp, \text{CA}^\perp, \text{DL}^\perp, \text{HTID} \otimes \text{HAID} \otimes \text{HDE} \)

- **Single input:**

  Buyer: \( \vdash \text{HDE}^\perp, \text{HO} \)
Example sub-proof

- **Multiple outputs:**
  
  HouseAlert: $\vdash HL^\perp, CA^\perp, DL^\perp, \ HTID \otimes HAID \otimes HDE$

- **Single input:**
  
  Buyer: $\vdash HDE^\perp, HO$

- How is HDE extracted from the composite output?
- What happens to HTID and HAID?
  - Buffering
Example sub-proof

- Need a formal proof! – nothing ad-hoc

- Multiple outputs:

\[ \text{HouseAlert: } \vdash \text{HL}^\perp, \text{CA}^\perp, \text{DL}^\perp, \text{HTID} \otimes \text{HAID} \otimes \text{HDE} \]
Handling exceptions

- **MortgageService may throw EXM:**
  
  \[
  \text{MortgageService: } \vdash \text{Cl}^\perp, \text{PA} \oplus \text{EXM}
  \]

- **ContractService can only handle PreApproval:**
  
  \[
  \text{ContractService: } \vdash \text{PA}^\perp, \text{AO}^\perp, \text{CO}
  \]
Handling exceptions

- MortgageService may throw EXM:
  \[ \text{MortgageService: } \vdash \text{CI} \downarrow, \text{PA} \oplus \text{EXM} \]

- ContractService can only handle PreApproval:
  \[ \text{ContractService: } \vdash \text{PA} \downarrow, \text{AO} \downarrow, \text{CO} \]

- How is PA extracted from the optional output?
- What happens if EXM is thrown?
  - Exception handling / tracking
Handling exceptions

\[
\begin{align*}
\vdash CI^\perp, PA \oplus EXM & \quad \textit{MortgageService} \quad \vdash AO^\perp, AO \\
\vdash CI^\perp, AO^\perp, (PA \oplus EXM) \otimes AO & \quad \vdash AO^\perp, AO \\
\vdash CI^\perp, AO^\perp, (PA \otimes AO) \oplus (EXM \otimes AO) & \quad \textit{DistribR} \\
\vdash EXM^\perp, EXM & \quad \vdash AO^\perp, AO \\
\vdash EXM^\perp, AO^\perp, EXM \otimes AO & \quad \vdash AO^\perp, AO \\
\vdash (EXM \otimes AO)^\perp, EXM \otimes AO & \quad \textit{neg_eq} \\
\vdash (EXM \otimes AO)^\perp, (CO \oplus (EXM \otimes AO)) & \quad \oplus \\
\vdash (PA \perp, AO^\perp, CO & \quad \textit{ContractService} \\
\vdash (PA \perp \not\triangleright AO^\perp), CO & \quad \textit{neg_eq} \\
\vdash (PA \otimes AO)^\perp, CO & \quad \oplus \\
\vdash (PA \otimes AO)^\perp, (CO \oplus (EXM \otimes AO)) & \quad \oplus \\
\vdash (PA \otimes AO)^\perp \& (EXM \otimes AO)^\perp, (CO \oplus (EXM \otimes AO)) & \quad \textit{neg_eq} \\
\vdash ((PA \otimes AO) \oplus (EXM \otimes AO))^\perp, (CO \oplus (EXM \otimes AO)) & \quad \oplus \\
\vdash CI^\perp, AO^\perp, (CO \oplus (EXM \otimes AO)) & \quad \oplus
\end{align*}
\]
WS Composition using proofs-as-processes

Translate to CLL

Prove Requested Service

Extract $\pi$-calculus term

Realisation ...
WS Composition using proofs-as-processes

Translate to CLL

Prove Requested Service

Extract $\pi$-calculus term

Realisation Execution
PiVizTool

- Bog (2006)

- Visualisation of connections
- Animation of execution
- Empirical verification
WS Composition using proofs-as-processes

Translate to CLL

Prove Requested Service

Extract $\pi$-calculus term

Realisation Translation

Upcoming!
BPEL
OWL-S

29
Related Work

- Workflow techniques (eg. EFlow)
- AI Planning (eg. SWORD)

- Theorem proving:
  - Waldinger (2001): program synthesis in SNARK (FOL)
  - Lammermann (2002): structural synthesis in Java (IL)
  - Rao et al. (2006): proofs-as-processes in RAPS (ILL)
  - Current project: proofs-as-processes in HOL Light (CLL)
Future Work

- Translation
  - From/to: BPEL(?) OWL-S(?)

- Verification
  - Prove Safety - Deadlock-freedom

- Automation
  - Internal and External tools

- Further evaluation
  - More examples

- Modalities
  - More expressiveness
Conclusion

- Offline Web Services composition.
  - Formally verified:
    - Systematic, trusted, extensible
    - Based on the proofs-as-processes paradigm.
    - Using CLL + π-calculus.
    - Implemented in HOL Light
  - Pragmatic:
    - Non-trivial example of 10 WS
    - Proper exception handling
- Also a TP framework for theoretical investigation
- A lot of ideas and room for further development