Abstraction and Modular Verification of Services Using Symbolic Observation Graph (SOG)

Hanen OCHI
Motivation

Services invoked by an organization

Workflow

Web services

Loosely Coopled
Motivation
Motivation

Abstraction

Composition
Motivation

Abstraction

Composition

Compatibility criteria
Approach based on SOG

**Symbolic Observation Graph SOG**

- Initially introduced for checking event-based LTL/X properties
- Observation of only events occurring in the formula

- **SOG:**

  \[
  G(a \iff F b) \rightarrow \text{Obs}\{a, b\}
  \]

  - **Hybrid Graph**
    - Nodes *(aggregates)*: sets of explicit states
      - Symbolic encoding *(BDDs)*
      - Symbolic algorithms *(deadlock, livelock)*
    - Edges: labelled by observed events
Approach based on SOG

• **Symbolic Observation Graph SOG**
  ✓ Initially introduced for checking event-based LTL/ X properties
  ✓ Observation of only events occurring in the formula
  • **SOG:**
    ➢ **Hybrid Graph**
      ✓ Nodes (aggregates): sets of explicit states
        • Symbolic encoding (BDDs)
        • Symbolic algorithms (deadlock, livelock)
      ✓ Edges: labelled by observed events
Approach based on SOG

• Symbolic Observation Graph SOG
  ✓ Initially introduced for checking event-based LTL/ X properties
  ✓ Observation of only events occurring in the formula

• SOG:
  ➢ Hybrid Graph
    ✓ Nodes (aggregates): sets of explicit states
      • Symbolic encoding (BDDs)
      • Symbolic algorithms (deadlock, livelock)
    ✓ Edges: labelled by observed events
Approach based on SOG

**Symbolic Observation Graph SOG**

- Initially introduced for checking event-based LTL/ X properties
- Observation of only events occurring in the formula

- **SOG:**
  - Hybrid Graph
    - Nodes (aggregates): sets of explicit states
      - Symbolic encoding (BDDs)
      - Symbolic algorithms (deadlock, livelock)
    - Edges: labelled by observed events

---

![Diagram of SOG](image)
Approach based on SOG

• Symbolic Observation Graph SOG

✓ Abstraction of the behavior of service
✓ Observation of only collaborative actions
   Actions of the collaboration: \textit{Obs}
   Actions of the internal behavior: \textit{UnObs}
Our Bottom-Up approach

Figure 1. Schema of our approach
Approach based on SOG

• Composition of SOGs
Approach based on SOG

• Composition of SOGs

Interlock: Deadlock caused by the interaction
Approach based on SOG

• Observed behavior $\lambda$

(!) A set of sets of transitions with which we can leave the aggregate.

$\lambda(a_1): \{\{o_1\}, \{o_2\}, \{term\}\}$

$\lambda(a_2): \{\{o_1\}, \{o_2\}, \{term\}\}$

Interlock: Deadlock caused by the interaction.
Exemple of Web services

• **Formal representation**: oWF-nets

• **Example**: Online shop

```
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>p11</td>
<td>Submit_order</td>
</tr>
<tr>
<td>p12</td>
<td>choose_deliv</td>
</tr>
<tr>
<td>p13</td>
<td>accept_pay</td>
</tr>
<tr>
<td>p14</td>
<td>keep_info</td>
</tr>
<tr>
<td>p15</td>
<td>init_deliv</td>
</tr>
<tr>
<td>o1</td>
<td></td>
</tr>
</tbody>
</table>
```
Asynchronous Composition

Interface compatible

Client 1

Online Shop

20/11/14
Asynchronous Composition

Client 1

\( i_2 \)

Verif\_art

\( N_1 \)

Submit\_order

\( p_{21} \)

\( p_{22} \)

pay\_bill

\( p_{23} \)

wait\_deliv

\( o_1 \)

Medium Net

\( N_2 \)

Submit\_order

order

payment

accept\_bill

delivery

init\_deliv

wait\_deliv

init\_deliv

Order

Online Shop

\( i_1 \)

Verif\_art

\( p_{11} \)

Submit\_order

order

payment

accept\_pay

keep\_info

\( p_{13} \)

\( p_{14} \)

\( p_{15} \)

\( o_1 \)

20/11/14
SOGs (Example)

SOG of onlineshop:

- \( A_0 \) \( \lambda = \{\text{submit}_{\text{order}}\} \)
- \( A_1 \) \( \lambda = \{\text{accept}_{\text{pay}}\} \)
- \( A_2 \) \( \lambda = \{\text{init}_{\text{deliv}}\} \)
- \( A_3 \) \( \lambda = \{\text{term}\} \)

SOG of client 1:

- \( A'_0 \) \( \lambda = \{\text{submit}_{\text{order}}\} \)
- \( A'_1 \) \( \lambda = \{\text{pay}_{\text{bill}}\} \)
- \( A'_2 \) \( \lambda = \{\text{wait}_{\text{deliv}}\} \)
- \( A'_3 \) \( \lambda = \{\text{term}\} \)

SOG of client 2:

- \( A''_0 \) \( \lambda = \{\text{submit}_{\text{order}}\} \)
- \( A''_1 \) \( \lambda = \{\text{wait}_{\text{deliv}}\} \)
- \( A''_2 \) \( \lambda = \{\text{pay}_{\text{bill}}\} \)
- \( A''_3 \) \( \lambda = \{\text{term}\} \)
Asynchronous composition using the medium service
Composition of SOGs

• Asynchronous composition using the medium service

\[ \lambda_0 = \{\text{submit}\_\text{order}, \text{pay}\_\text{bill}, \text{init}\_\text{deliv}\} \]

\[ \lambda_{A0} = \{\text{submit}\_\text{order}\} \]

\[ \lambda_{A'0} = \{\text{submit}\_\text{order}\} \]

\[ \lambda = \{\text{submit}\_\text{order}\} \]

Initial product aggregate composed by initial aggregates of the onlineshop’s SOG, the medium net’s SOG and the client’s SOG
Composition of SOGs

•Theorem:
  Let $\mathcal{WS}$ a web service and let $\mathcal{G}$ the associated SOG
  $\mathcal{WS}$ is deadlockfree $\iff$ $\mathcal{G}$ is deadlockfree

Theorem:
Let $N_1$ and $N_2$ be two oWF-nets and let $G_1$ and $G_2$ be the corresponding SOGs respectively:
The composition of two SOGs $(G_1, \text{Obs}_1)$ and $(G_2, \text{Obs}_2)$, denoted $G_1 \oplus G_2$, is a SOG of $N_1 \ominus N_2$
Composition

Synchronized product of SOGs (with client 1)

Synchronized product of SOGs (with client 2)
Checking Compatibility

Synchronized product of SOGs (with client 1)

\[ A_0, 0, A' \]
\[ \lambda = \{ \text{submit order} \} \]

\[ A_1, 4, A' \]
\[ \lambda = \{ \text{pay bill} \} \]

\[ A_2, 2, A' \]
\[ \lambda = \{ \text{wait deliv} \} \]

\[ A_3, 1, A' \]
\[ \lambda = \{ \text{term} \} \]

Synchronized product of SOGs (with client 2)

\[ A_0, 0, A'' \]
\[ \lambda = \{ \text{submit order} \} \]

\[ A_1, 4, A'' \]
\[ \lambda = \{ \emptyset \} \]

**Theorem**: Deadlock freeness
A SOG \( G \) is said to be deadlock free \( \iff \nexists a \in G \mid \emptyset \in a.\lambda \)
Some properties:

**Generic properties:**

**Deadlockfreeness**

**Soundness:**
- Option to complete
- Proper completion
- No dead transitions

- **Relaxed Soundness**
  - Each transition occurs in at least one “good” execution path.
- **Weak Soundness**
  - A final marking is reachable from any reachable state.
- **Easy Soundness**
  - A final marking is reachable from the initial marking.

**Specific properties:**

*Properties expressed with LTL*

Enrich aggregates with locally computed information !!!
## Experimental Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Nom</th>
<th>Sound</th>
<th>Places</th>
<th>Trans</th>
<th>Obs</th>
<th>S</th>
<th>E</th>
<th>T(s)</th>
<th>Obs</th>
<th>S</th>
<th>E</th>
<th>T(s)</th>
<th>LoLA S</th>
<th>E</th>
<th>T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C+SC1</td>
<td>C+SC1</td>
<td>F</td>
<td>25</td>
<td>16</td>
<td>4/4</td>
<td>9</td>
<td>8</td>
<td>&lt;1</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>&lt;1</td>
<td>22</td>
<td>26</td>
<td>&lt;1</td>
</tr>
<tr>
<td>C+SC2</td>
<td>C+SC2</td>
<td>T</td>
<td>28</td>
<td>23</td>
<td>4/4</td>
<td>11</td>
<td>12</td>
<td>&lt;1</td>
<td>8</td>
<td>11</td>
<td>12</td>
<td>&lt;1</td>
<td>23</td>
<td>25</td>
<td>&lt;1</td>
</tr>
<tr>
<td>Res</td>
<td>Res</td>
<td>T</td>
<td>28</td>
<td>33</td>
<td>8/8</td>
<td>17</td>
<td>17</td>
<td>&lt;1</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>&lt;1</td>
<td>19</td>
<td>21</td>
<td>&lt;1</td>
</tr>
<tr>
<td>ResTrip</td>
<td>ResTrip</td>
<td>F</td>
<td>30</td>
<td>26</td>
<td>7/5</td>
<td>10</td>
<td>12</td>
<td>&lt;1</td>
<td>12</td>
<td>10</td>
<td>12</td>
<td>&lt;1</td>
<td>24</td>
<td>27</td>
<td>&lt;1</td>
</tr>
</tbody>
</table>
Conclusion & Further work

- Study of some approaches for abstraction web services

- New version of the graph of symbolic observation adapted to services
  - And business processes

- Checking for compatibility based on Soundness and its variant

- Implementation:
  - Deadlock-freeness (integrated to CosyVerif)
  - Soundness variants
  - LTL modular model checking (loading ....)

- Further work:
  - Consider shared resources
  - Consider time explicitly
Conclusion & Further work

- Study of some approaches for abstraction web services
- New version of the graph of symbolic observation adapted to services
  - And business processes
- Checking for compatibility based on Soundness and its variant
- Implementation:
  - Deadlock-freeness (integrated to CosyVerif)
  - Soundness variants
  - LTL modular model checking (loading ....)
- Further work:
  - Consider shared resources
  - Consider time explicitly
Thank you for your attention
Thank you for your attention
Revisiting SOGs for modular verification of soundness properties

- $M_f(a) = \{m \in S \mid \exists m_f \in \Omega: m_f \in R(N, m)\}$ is a set of marking;
- $\text{Enable}(a.S) = \{t \in T \mid \exists m \in S: m \rightarrow^t\}$ is a set of transitions;

$a = \langle S, \lambda, M_f, \text{Enable} \rangle$
Revisiting SOGs for modular verification of soundness properties

- \( M_f(a) = \{m \in S \mid \exists m_f \in \Omega: m_f \in R(N, m)\} \) is a set of marking;
- \( \text{Enable}(a.S) = \{t \in T \mid \exists m \in S: m \rightarrow^t\} \) is a set of transitions;

Let \( G = \langle A, \text{Act}, \rightarrow, a_0, \Omega' \rangle \) be a SOG
- option to complete: \( \bigcup_{a \in A} M_f(a) = \bigcup_{a \in A} a.S \).
- proper completion: \( \forall a \in A, \forall m \in a.S, \forall m_f \in \Omega, m \geq m_f \Rightarrow m = m_f; \)
- no dead transitions: \( \bigcup_{a \in A} \text{Enable}(a.S) = T. \)
Revisiting SOGs for modular verification of soundness properties

The Modular Verification:

\[ Enable(a) = \bigcup_{i=1,2} (\text{Enable}(a_i) \setminus (\text{Obs}_i \cap \text{Obs}_{12})) \cup (\text{Enable}(a_i) \cap \text{Enable}(a_{12})) \]

**Theorem:**

\[ G_i = \langle \mathcal{A}_i, a_0, \rightarrow_i, \mathcal{F}_i \rangle \quad (i = 1, 2) \text{be two SOGs corresponding to } \mathcal{N}_1 \text{ and } \mathcal{N}_2 \]

- if \( G_1 \) and \( G_2 \) are sound then \( G_1 \oplus G_2 \) is sound if the following requirements are satisfied:
  - option to complete: \( \forall a \in \mathcal{A}, \emptyset \not\in a.\lambda \land \exists a_f \in \mathcal{F} \mid a_f \in R(a) \).
  - no dead transitions: \( \bigcup_{a \in \mathcal{A}} \text{Enable}(a.S) = \bigcup_{i=1,2} \bigcup_{a_i \in \mathcal{A}_i} \text{Enable}(a_i.S) \).
Revisiting SOGs for modular verification of relaxed soundness properties

\[ T_f(a) = \{ t \in T \mid \text{Succ}(M_f(a), t) \cap M_f(a) \neq \emptyset \} \] a set of transitions \( \text{Succ}(S, t) = \{ s' \mid \exists s \in S: s \xrightarrow{t} s' \} \) the set of states reachable from any state of \( S \) by the firing of \( t \).

\[ a = \langle S, \lambda, M_f, T_f \rangle \]
Revisiting SOGs for modular verification of relaxed soundness properties

\[ T_f(a) = \{ t \in T \mid \text{Succ}(M_f(a), t) \cap M_f(a) \neq \emptyset \} \] a set of transitions \[ \text{Succ}(S, t) = \{ s' \mid \exists s \in S : s \xrightarrow{t} s' \} \] the set of states reachable from any state of \( S \) by the firing of \( t \).

\[ a = \langle S, \lambda, M_f, T_f \rangle \]

**Theorem:**

Let \( G = \langle A, Act, \rightarrow, a_0, \Omega' \rangle \) be a SOG associated with an oWF-net \( N \)

- \( N \) is relaxed sound iff \( \bigcup_{a \in A} T_f(a) = T \).
Revisiting SOGs for modular verification of relaxed soundness properties

The Modular Verification:

\[ T_f(a) = \bigcup_{i=1}^{2} (T_f(a_i) \setminus (\text{Obs}_i \cap \text{Obs}_{12})) \cup (T_f(a_i) \cap T_f(a_{12})) \]

• Theorem:

\( \mathcal{G}_i = \langle \mathcal{A}_i, a_{0i}, \rightarrow_i, \mathcal{F}_i \rangle \) (\( i = 1, 2 \)) be two SOGs corresponding to \( \mathcal{N}_1 \) and \( \mathcal{N}_2 \)

- if \( \mathcal{G}_1 \) and \( \mathcal{G}_2 \) are relaxed sound and \( \mathcal{G}_1 \oplus \mathcal{G}_2 \) does not contain composed deadlocks, then \( \mathcal{G}_1 \oplus \mathcal{G}_2 \) is relaxed sound if \( \forall t \in \text{Obs}_1 \cup \text{Obs}_2 \exist a \in \mathcal{A} \exist a_f \in \mathcal{F} : t \in T_f(a) \land a_f \in R(a) \) (i.e. \( \bigcup_{a \in \mathcal{A}} T_f(a) = \text{Obs}_1 \cup \text{Obs}_2 \))
Revisiting SOGs for modular verification of weak and easy soundness properties

Theorem:

Let $G = \langle \mathcal{A}, \text{Act}, \rightarrow, a_0, \Omega' \rangle$ be a SOG associated with an oWF-net $N$

- $N$ is weak sound iff $\bigcup_{a \in \mathcal{A}} M_f(a) = \bigcup_{a \in \mathcal{A}} a.S$
- $N$ is easy sound iff $\bigcup_{a \in \mathcal{A}} M_f(a) \neq \emptyset$

Theorem:

if $G_1$ and $G_2$ are weak sound then $G_1 \oplus G_2$ is weak sound $\forall a \in \mathcal{A}$, $\emptyset \notin a.\lambda \land \exists a_f \in F \mid a_f \in R(a)$.

$G_1 \oplus G_2$ is easy sound if $R(a_0) \cap F \neq \emptyset$. 

20/11/14