Transforming BPEL into Petri Nets

Sebastian Hinz, Karsten Schmidt, Christian Stahl
Motivation

BPEL:

• modelling language for business processes
• BPEL process can be very huge
→ formal verification of BPEL processes needed

problems:

• BPEL is specified informally
• formal verification of BPEL processes not possible
Roadmap

1. Petri net semantics for BPEL
2. Automatic transformation
3. Computer-aided verification of BPEL processes
4. Improved transformation
1. Petri net semantics for BPEL
Process

Sequence

Flow

Switch

Fault Handler

Compensation Handler
Idea of the Petri net semantics

• complete Petri net semantics for BPEL v1.1

Sequence

Flow

• Petri net patterns
• interface
Example: receive
Controlled stop of a process

- process is forced to stop, e.g. when fault occurs
- specification cloudy
- modelling decisions:
  - process is extended by a stop pattern
  - stop pattern controls the stop procedure
  - removes all tokens
  - each activity pattern is extended by a stop component
Example: receive (cont.)

extend the interface
2. Automatic transformation
Schema of transformation

- transformation can be done automatically → compiler $BPEL2PN$
- abstraction from data
Example: receive

... abstraction from data
3. Computer-aided verification of BPEL processes
Schema

- BPEL process
- Petri net semantics
- Model Checker
- analyse
- properties
Validation and Analysis

- Validation of the semantics
- Analysis of BPEL processes

Model Checking
### Analysis results

<table>
<thead>
<tr>
<th></th>
<th>PO</th>
<th>Shop</th>
<th>Bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>activities</td>
<td>17</td>
<td>53</td>
<td>132</td>
</tr>
<tr>
<td>places</td>
<td>158</td>
<td>410</td>
<td>1,890</td>
</tr>
<tr>
<td>transitions</td>
<td>249</td>
<td>1,069</td>
<td>21,193</td>
</tr>
<tr>
<td>states</td>
<td>10,000</td>
<td>6,300,000</td>
<td>?</td>
</tr>
<tr>
<td>red. states</td>
<td>1,300</td>
<td>440,000</td>
<td>?</td>
</tr>
</tbody>
</table>

using the model checker LoLA

**problem:**

- real-life processes cause **state-explosion**
- smaller models needed
4. Improved transformation
Improved transformation

- so far patterns applicable in every context
- many of the modelled features are unused

solution:

1. get knowledge of an activity’s context
   - Can an activity throw a fault?
2. derive aspects needed to prove a property
   - external behaviour: communication only
receive: external behaviour
Flexible model generation

for each BPEL activity several patterns with different degree of abstraction

- enhanced Petri net semantics
- Model Checker

BPEL process

transform

analyse

properties

information, behaviour

static analysis
Further Work

• flexible model generation
• extend the semantics to BPEL v2.0
  – termination handler
  – ...

Many thanks