Checking Business Process Evolution

Gwen Salaün
Université Grenoble Alpes, LIG, Inria, France

joint work with
Ajay Krishna and Pascal Poizat
Introduction (1/2)

- A business process is a set of structured, related activities or tasks designed to produce a specific output for a customer or market.

- BPMN 2.0 (Business Process Modelling Notation) was published as an ISO/IEC standard in 2013.

- Modern software exhibits a high degree of dynamicity and is subject to continuous evolution.
Given two BPMN business processes, we want to support the process designer in the evolution activity with automated verification techniques.

Formal modelling and analysis is important and required to ensure correctness, efficiency, and quality of the whole process execution.

Our contributions:
- An LTS (Labelled Transition System) semantics for a subset of BPMN obtained via process algebra encoding
- A set of evolution notions based on LTS equivalences / preorders that one can use to compare two processes
- A tool support for fully automating the evolution checks that can be accessed via a Web application
Outline

1. BPMN
2. From BPMN to LTS
3. Process Comparison
4. Tool Support
5. Concluding Remarks
Several modelling and development frameworks: Activiti, Bonita BPMN, jBPM, …
Control Flows and Gateways

- Sequence flow
- Start state
- End state
- Exclusive gateway
- Event-based gateway
- Inclusive gateway
- Parallel gateway
- Split pattern
- Merge pattern
Semantics

Exclusive gateway (similar for Event-based gateway): split (left) and merge (right)

Parallel gateway: split (left) and merge (right)

Inclusive gateway: split (left) and merge (right)
Example of BPMN Process

Bank account opening process
Outline

1. BPMN
2. From BPMN to LTS
3. Process Comparison
4. Tool Support
5. Concluding Remarks
LNT

- LOTOS NT (LNT) is a value-passing process algebra with user-friendly syntax and operational semantics

- LNT is an imperative-like language where you can specify data types, functions (pattern matching and recursion), and processes

- Excerpt of the LNT process grammar:

  \[
  B ::= \text{stop} \mid G(!E, ?X) \text{ where } E' \mid \text{if } E \text{ then } B1 \text{ else } B2 \text{ end if} \\
  \mid x := E \mid \text{hide } G \text{ in } B \text{ end hide} \mid P [G1, \ldots, Gm] (E1, \ldots, En) \\
  \mid \text{select } B1 [\ldots] Bn \text{ end select} \mid B1 ; B2 \\
  \mid \text{par } G \text{ in } B1 || \ldots || Bn \text{ end par}
  \]

- Compilation to LTS and verification using the CADP toolbox
<table>
<thead>
<tr>
<th>BPMN construct</th>
<th>BPMN notation</th>
<th>LNT encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial event</td>
<td><img src="image" alt="Initial event" /></td>
<td>begin ; outf</td>
</tr>
<tr>
<td>End event</td>
<td><img src="image" alt="End event" /></td>
<td>incf ; finish</td>
</tr>
<tr>
<td>Sequence flow</td>
<td><img src="image" alt="Sequence flow" /></td>
<td>loop begin ; finish end loop</td>
</tr>
<tr>
<td>Task</td>
<td><img src="image" alt="Task" /></td>
<td>loop incf ; task ; outf end loop</td>
</tr>
<tr>
<td>Parallel gateway (split)</td>
<td>![Parallel gateway (split)]</td>
<td>incf ; par</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outf1</td>
</tr>
<tr>
<td>Parallel gateway (merge)</td>
<td>![Parallel gateway (merge)]</td>
<td>par</td>
</tr>
<tr>
<td></td>
<td></td>
<td>incf1</td>
</tr>
<tr>
<td>Exclusive gateway (split)</td>
<td>![Exclusive gateway (split)]</td>
<td>incf ; select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outf1</td>
</tr>
<tr>
<td>Exclusive gateway (merge)</td>
<td>![Exclusive gateway (merge)]</td>
<td>select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>incf1</td>
</tr>
<tr>
<td>Inclusive gateway (split)</td>
<td>![Inclusive gateway (split)]</td>
<td>incf ; select (* si if one matching merge *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outf1 ; s1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>outf2 ; s2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>par outf1</td>
</tr>
<tr>
<td>Inclusive gateway (merge)</td>
<td>![Inclusive gateway (merge)]</td>
<td>select (* si if one matching split *)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s1 ; incf1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s2 ; incf2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>s3 ; par incf1</td>
</tr>
</tbody>
</table>
LNT Encoding (2/2)

```plaintext
process main [processApplication:any, reply:any, createProfile:any, ...] is
hide begin:any, finish:any, flow1_begin:any, flow1_finish:any, ... in
par flow1_begin, flow1_finish, flow2_begin, flow2_finish, ... in
par
  flow [flow1_begin, flow1_finish] ||...|| flow [flow29_begin, flow29_finish]
end par
||
par
  init [begin,flow1_begin]
|| final [flow21_finish, finish] || final [flow27_finish, finish]
|| task [flow1_finish, processApplication, flow2_begin] || task [...] || ...
|| xorsplit [flow2_finish, flow3_begin, flow4_begin]
|| xormerge [flow6_finish, flow7_finish, flow29_begin]
end par
end par
end hide
end process
```
Outline

1. BPMN
2. From BPMN to LTS
3. Process Comparison
4. Tool Support
5. Concluding Remarks
Given two BPMN processes $P_1$ and $P_2$, we can define several notions of comparison using concurrency theory:

- **Conservative evolution**: both processes exhibit exactly the same behaviour. 
  \[ \text{LTS}(P_1) =_{br} \text{LTS}(P_2) \]

- **Inclusive / exclusive evolution**: one process simulated by the other. 
  \[ \text{LTS}(P_1) <_{br} \text{LTS}(P_2) \quad (\text{LTS}(P_1) >_{br} \text{LTS}(P_2), \text{resp.}) \]

- **Up-to-alphabet / up-to-renaming evolution**: checking former relations hiding or renaming parts of the alphabet. 
  \[ \text{LTS}'(P_1) =_{br} \text{LTS}'(P_2) \text{ where } \text{LTS}'(P_i) = \text{hide } A \text{ in } \text{LTS}(P_i) \]

- **Property preserving evolution**: both processes satisfy a same temporal property $P$. 
  \[ \text{LTS}(P_1) \models P \text{ and } \text{LTS}(P_2) \models P \]
Conservative evolution ✗
Inclusive evolution ✓
Exclusive evolution ✗
Property preserving evolution « any process execution eventually terminates by a rejection notification or by an account activation » ✗
Outline

1. BPMN
2. From BPMN to LTS
3. Process Comparison
4. Tool Support
5. Concluding Remarks
The VBPMN Platform

BPMN 2.0 compliant platforms
(Activiti, Bonita BPM, ..)

Other workflow-based languages
(UML activ. diagrams, YAWL, ..)

Process Intermediate Format

Generated code
(LNT + SVL)

CADP verification tools

Model-to-model Transformation

PIF

Model-to-text Transformation

Other verification techniques
(formal proof, testing, ..)

Diagnostics

https://pascalpoizat.github.io/vbpmn/
Process Intermediate Format

- BPMN
- CADP
- UML Activity Diagrams
- UPPAAL
- Maude
- YAWL
- New tools
Web Interface

VBPMN: BPMN Model Comparison
Service for comparison and analysis of BPMN 2.0 models

Request form

File input 1
[Choose File] PublishingSystemV3.bpmn
BPMN 2.0 model with file input

File input 2
[Choose File] PublishingSystemV2.bpmn
BPMN 2.0 model with file input

Mode
conservative

Options
☐ None  ☐ Hide  ☐ Rename

Submit
## Experiments

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tasks</td>
<td>Flows</td>
<td>Gateways</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>11</td>
<td>2 [x]</td>
</tr>
<tr>
<td>1’</td>
<td>7</td>
<td>15</td>
<td>2 [x] + 2 [+]</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>7</td>
<td>1 [○]</td>
</tr>
<tr>
<td>2’</td>
<td>8</td>
<td>14</td>
<td>2 [x]</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>14</td>
<td>2 [x] + 2 [+]</td>
</tr>
<tr>
<td>3’</td>
<td>8</td>
<td>16</td>
<td>4 [○]</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>29</td>
<td>3 [x] + 2 [+] + 2 [○]</td>
</tr>
<tr>
<td>4’</td>
<td>16</td>
<td>33</td>
<td>5 [x] + 2 [+] + 2 [○]</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>24</td>
<td>6 [○]</td>
</tr>
<tr>
<td>5’</td>
<td>12</td>
<td>24</td>
<td>4 [○] + 2 [x]</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>43</td>
<td>6 [○] + 6 [+]</td>
</tr>
<tr>
<td>6’</td>
<td>20</td>
<td>39</td>
<td>8 [○]</td>
</tr>
</tbody>
</table>
Outline

1. BPMN
2. From BPMN to LTS
3. Process Comparison
4. Tool Support
5. Concluding Remarks
Concluding Remarks

- We have presented an approach for automatically checking the evolution of BPMN processes.
- We have defined a BPMN to LNT translation, which allows to provide an LTS semantics for BPMN.
- We have proposed several notions of evolution taking inspiration in concurrency theory.
- We have implemented our approach in a tool, VBPMN, which can be used via a Web application.

Perspectives: support of unbalanced workflows, extending the BPMN subset considered, quantitative analysis, …