Semantic Business Process Validation

Ingo Weber, Jörg Hoffmann
SAP Research, Karlsruhe, Germany

Jan Mendling
Queensland University of Technology, Australia
1. Introduction
2. Formalism
3. Algorithms
4. Conclusion
In a Nutshell

- **Traditional BPs**
  - Workflow (AND/XOR splits/joins)
  - Validation checks workflow soundness

- **Semantic BPs**
  - Domain ontology formalizes the domain which the tasks in the process affect
  - Each task annotated with precondition and effect
  - Formal execution semantics combines workflow with AI actions & change

- **Semantic BP Validation**
  - Workflow and annotations interact!
  - Is there a precondition that may be violated?
  - Are there parallel (un-ordered) tasks with contradicting effects?
  - Useful to debug high-level models and/or SWS implementations
An Example

Drafted \text{SubClassOf} Prepared
Updated \text{SubClassOf} Prepared
Completed \text{DisjointClasses} Prepared

Drafted \text{calc} 
Prepared \text{calc} 
Updated \text{calc} 
Completed \text{calc} 

Decision on Shipper
SBPM Branches of Related Work

- **SUPER [Hepp et al]**
  - Our project
  - Ontology (here) models the domain in which the process is executed

- **TOVE [Fox]**
  - Ontologies for modeling enterprises and processes
  - Our work is mostly on the Generic Model Level of TOVE
  - PSL (Process Specification Language) talks about tasks, time points, and constraints between tasks; no “execution in the presence of domain ontology”

- **DEMO [Dietz]**
  - Method to discover/mine the processes executed within an enterprise
  - Essentially unrelated to our work (except a “process” occurs somewhere)
1. Introduction
2. Formalism
3. Algorithms
4. Conclusion
Semantic Business Processes

- **Process Graph**
  - Traditional workflow
  - Nodes: start/end, task, AND split/join, XOR split/join
  - Simple token-passing semantics (borrowed from [Vanhatalo et al ICSOC’07])

- **Semantic Annotation**
  - Ontology: universally quantified clauses -- $\forall x: \text{Drafted}(x) \Rightarrow \text{Prepared}(x)$
  - Task node precondition/effect: conjunctions of literals -- $\text{Drafted}(\text{calc})$

- **Execution Semantics**
  - Combines token passing with AI “minimal change semantics” [Winslett AAAI’89]
  - In the presence of ontology $\Omega$, if $\phi$ happens in state $s$, then any state $s'$ may result that satisfies $\phi$ and $\Omega$, and that differs minimally from $s$
  - “From the annotations, we can not always conclude with certainty what will happen; we assume that nothing changes without a reason”
Non-Executable Tasks

Drafted SubClassOf Prepared
Updated SubClassOf Prepared
Completed DisjointClasses Prepared

Decide on Shipper
Effect Conflicts

Drafted (calc) SubClassOf Prepared
Updated (calc) SubClassOf Prepared
Completed DisjointClasses Prepared

Drafted

Updated

Completed
Outline

1. Introduction
2. Formalism
3. Algorithms
4. Conclusion
In a Nutshell

- **Semantic BP Validation**
  - Are there non-executable tasks?
  - Are there effect conflicts?

- **Computational Complexity**
  - Validation is NP-hard if XOR splits have conditions
  - Validation is NP-hard if ontology has Horn clauses
  - Open question for loops

- **Basic Semantic BPs**
  - No XOR conditions, only 2-clauses, no loops
  - E.g. the example
  - Polynomial-time validation techniques
  - Extension for loops currently investigated
“Matrix”
- Determines “parallel task nodes”
- Propagates entries of a $|E| \times |E|$ matrix $M$
- Upon completion, task nodes with ingoing edges $i, j$ parallel iff $M_{ij} = 1$

“I-Propagation”
- Determines “necessarily true literals”
- Propagates sets $I(e)$ annotated at edges $e$
  - AND/XOR joins propagate the union/intersection of their ingoing edges
  - Interference from parallel task node effects must be considered
  - 2-clauses can be “compiled into” extended preconditions & effects
- Upon completion, $I(e)$ contains exactly the set of literals that are true whenever edge $e$ carries a token
I-Propagation – Example (1)

I = \{\text{orderReceived}(o), \neg \text{orderCompleted}(o)\}
I-Propagation – Example (2)

I = \{\text{orderReceived}(o), \neg \text{orderCompleted}(o)\}

Receive Order

Production Scheduling

Decide on Shipper

Arrange Logistics
I-Propagation – Example (3)

I = \{orderReceived(o), \neg orderCompleted(o)\}

Receive Order

I = \{orderReceived(o), \neg orderCompleted(o), shipperDecided(o, s)\}

Production Scheduling

Decide on Shipper

Arrange Logistics
1. Introduction
2. Formalism
3. Algorithms
4. Conclusion
In a Nutshell

Semantic BP Validation
- Requires ontology formalizing the domain affected by process activities
- Validates interaction between semantic annotation and workflow
- Helps with high-level modeling
- Helps resolving implementation bugs in case of Semantic WS

Contribution
- Formal execution semantics combines workflow and AI actions&change
- (Almost) maximal tractable class: basic semantic BPs

Future Work
- Loops?
- Empirical evaluation
- Validation for more general (intractable) cases
Thank you!

Partially an outcome of EU IP SUPER
http://www.ip-super.org
Backup Slides
Semantic Process Validation

example process
### Semantic Process Validation

#### example preconditions / postconditions

<table>
<thead>
<tr>
<th>Task</th>
<th>Precondition</th>
<th>Postcondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive Order</td>
<td>orderReceived(o)</td>
<td>orderReceived(o)</td>
</tr>
<tr>
<td>Production Scheduling</td>
<td>orderReceived(o) orderApproved(o)</td>
<td>productionScheduled(o,p)</td>
</tr>
<tr>
<td>Production</td>
<td>productionScheduled(o,p)</td>
<td>productionCompleted(o,p)</td>
</tr>
<tr>
<td></td>
<td>calculationPrepared(o,c)</td>
<td>calculationUpdated(o,c)</td>
</tr>
<tr>
<td>Decide on Shipper</td>
<td>orderReceived(o)</td>
<td>shipperDecided(o,s)</td>
</tr>
<tr>
<td>Arrange Logistics</td>
<td>shipperDecided(o,s)</td>
<td>calculationUpdated(o,c)</td>
</tr>
<tr>
<td></td>
<td>calculationPrepared(o,c)</td>
<td>shipmentApproved(o,sh)</td>
</tr>
<tr>
<td>Draft Price Calculation</td>
<td>orderReceived(o)</td>
<td>calculationDrafted(o,c)</td>
</tr>
<tr>
<td>Complete Price Calculation</td>
<td>calculationPrepared(o,c)</td>
<td>calculationCompleted(o,c)</td>
</tr>
<tr>
<td>Invoice Processing</td>
<td>productionCompleted(o,p)</td>
<td>orderCompleted(o)</td>
</tr>
<tr>
<td></td>
<td>calculationCompleted(o,c)</td>
<td></td>
</tr>
</tbody>
</table>
# Semantic Process Validation

## example ontology

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order status</td>
<td>$o$ is at most one of <em>received</em> or <em>completed</em></td>
</tr>
<tr>
<td>Production status</td>
<td>$p$ is at most one of <em>scheduled</em> or <em>completed</em></td>
</tr>
<tr>
<td>Calculation status</td>
<td>if $c$ is <em>drafted</em>, then $c$ is <em>prepared</em></td>
</tr>
<tr>
<td></td>
<td>if $c$ is <em>updated</em>, then $c$ is <em>prepared</em></td>
</tr>
<tr>
<td></td>
<td>$c$ is at most one of <em>drafted</em> or <em>updated</em> or <em>completed</em></td>
</tr>
<tr>
<td></td>
<td>$c$ is at most one of <em>prepared</em> or <em>completed</em></td>
</tr>
<tr>
<td>Order approval</td>
<td>if shipment $sh$ is <em>approved</em> for $o$, then $o$ is <em>approved</em></td>
</tr>
</tbody>
</table>
I-Propagation – Example (1)

I = \{\text{orderReceived}(o), \neg\text{orderCompleted}(o)\}

Receive Order

Production Scheduling

Decide on Shipper

Arrange Logistics
I-Propagation – Example (2)

I = \{\text{orderReceived}(o), \neg \text{orderCompleted}(o)\}

Receive Order

Decide on Shipper

Arrange Logistics

Production Scheduling
I-Propagation – Example (3)

I = \{\text{orderReceived}(o), \neg\text{orderCompleted}(o)\}

Receive Order

Production Scheduling

Decide on Shipper

I = \{\text{orderReceived}(o), \neg\text{orderCompleted}(o), \text{shipperDecided}(o,s)\}

Arrange Logistics
I-Propagation – Example (4)

I = \{orderReceived(o), orderCompleted(o), productionScheduled(o,p), productionCompleted(o,p)\}

Receive Order

I = \{orderReceived(o), ¬orderCompleted(o)\}

Decide on Shipper

I = \{orderReceived(o), ¬orderCompleted(o), shipperDecided(o,s)\}

Arrange Logistics

I = \{orderReceived(o), ¬orderCompleted(o)\}

Production Scheduling
I-Propagation – Example (5)

- **I = {orderReceived(o), ¬orderCompleted(o)}**
- **I = {orderReceived(o), ¬orderCompleted(o), shipperDecided(o,s)}**
- **I = {orderReceived(o), ¬orderCompleted(o), productionScheduled(o,p), ¬productionCompleted(o,p)}**
- **I = {orderReceived(o), ¬orderCompleted(o), shipperDecided(o,s), calculationUpdated(o,c), calculationPrepared(o,c), ¬calculationDrafted(o,c), ¬calculationCompleted(o,c), shipmentApproved(o,sh), orderApproved(o)}**
I-Propagation – Example (6)

I = \{\text{orderReceived}(o), \neg\text{orderCompleted}(o), \text{productionScheduled}(o,p), \neg\text{productionCompleted}(o,p)\}

I = \{\text{orderReceived}(o), \neg\text{orderCompleted}(o), \text{shipperDecided}(o,s), \text{calculationUpdated}(o,c), \text{calculationPrepared}(o,c), \neg\text{calculationDrafted}(o,c), \neg\text{calculationCompleted}(o,c), \text{shipmentApproved}(o,sh), \text{orderApproved}(o), \text{productionScheduled}(o,p), \neg\text{productionCompleted}(o,p)\}

I = \{\text{orderReceived}(o), \neg\text{orderCompleted}(o), \text{shipperDecided}(o,s), \text{calculationUpdated}(o,c), \text{calculationPrepared}(o,c), \neg\text{calculationDrafted}(o,c), \neg\text{calculationCompleted}(o,c), \text{shipmentApproved}(o,sh), \text{orderApproved}(o)\}