Automatic Generation of Model-to-Model Transformations from Rule-Based Specifications of Operational Semantics

Hans Vangheluwe† and Juan de Lara‡

†School of Computer Science
McGill University, Montréal, Canada

‡Escuela Politécnica
Universidad Autónoma de Madrid, Madrid, Spain

7th OOPSLA Workshop
on Domain-Specific Modeling (DSM’07)
21 – 22 October 2007
Montréal, Québec, Canada
Overview

1. Domain-Specific Visual Modelling (Production System example)
2. Supporting Simulation and Analysis
3. Petri Nets
4. Production System to Petri Nets mapping
5. Conclusions, Current and Future Work
A Production System Model

bar -> cyl

oper.

cyl

bar -> bar

assem.

oper. -> assem.

quality

pack

rework

repair
Modelling Concrete Syntax (and UI Behaviour)
Modelling Operational Semantics in the form of Rules

assemble

Note the use of **concrete** syntax!
Modelling Operational Semantics in the form of Rules

assemble

move

change

Modelling DSVLs/DSVEs
Rule Control Structures (layers, priorities, ...)
More than simulation ...  

### Operational Semantics ... and analysis

- high expressiveness of graph transformation  
  ⇒ very hard to prove semantic properties
- alternative: express semantics in terms of a formalism suitable for analysis (such as Petri Nets)  
  ⇒ “denotational” semantics

### Problems

- no longer domain-specific concepts/notations/constraints
- domain-expert needs to learn Petri Nets
- domain-expert must understand (complex) mapping onto Petri Nets
- for realistic models, Petri Net becomes very complex
- need to prove that operational and denotational semantics are equivalent
Solution

- automatic transformation of
  1. Production System model
  2. Operational Semantics rules
  3. (rule) matching/rewriting Semantics
  4. rule Control Structure

into a Petri Net (for simulation/analysis)

Note

need “backward links” to relate simulation/analysis results back to Production System domain
Expressiveness of Place/Transition nets

Join

\[ p1 \quad t1 \quad p2 \]
Expressiveness of Place/Transition nets
Concurrency

```
 1  
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
p11

0  
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
t1

1  
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
p21

0  
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
</table>
t2

p12

p22
```
Expressiveness of Place/Transition nets

Conflict, choice, decision

Diagram:
- Place 1
- Transition t1
- Place 0
- Transition t2
- Place 0
- Transition t2
- Place 0
- Transition t1

Edges:
- p1 from 0 to t1
- p3 from 1 to t1
- p2 from 0 to t2
- p1 from 0 to t2
Application: Critical Section

```
1 0 1 0
process1  critical1  semaphore  critical2  process2

t1  t1e

t2  t2e
```
Reachability Graph

Allows checking that for all possible behaviours, the system can never be in critical1 and critical2 simultaneously.
Back to our proposed solution

automatic transformation of

1. Production System model
2. Operational Semantics rules
3. (rule) matching/rewriting Semantics
4. rule Control Structure

into a Petri Net (for simulation/analysis)
Meta-Triples linking Production System and Petri Nets
Mapping the Production System model

The mapping...
How: Transformation Triple-Rules

- **How**: Transformation Triple-Rules
- **Machine**
- **MachP**: type="operator" modifier=1
- **MachP**: type="operator" modifier=0
- **OpTok**: type="operator" modifier=1
- **ConvC**: type="cylinder" modifier=1
- **0-op-assem.**
- **1-op-assem.**
- **0-op-assem.**
- **1-cyl-conv.**
- **1-op-machine**
- **init 0-op-machine**
- **add 1-op-machine**
- **add 0-op-machine**
- **init 1-op-machine**
- **add 1-cyl-conv.**

- **NAC**
- **new**

- **oper.**
Mapping Operational Semantics rules
How: Transformation Triple-Rules
Mapping Rule Priorities

Diagram showing the mapping rule priorities with nodes and arrows representing the relationships between different rules and states.
Claims
- the sketched triple rules work as intended:
- the generated Petri Net mimics operational semantics
- these rules can be generated automatically (by rules)!

Main Contribution
You can Have Your Cake and Eat It!
- domain-expert-friendly specification of syntax and operational semantics
- and analyzability through automated mapping onto Place/Transition Petri Nets
A Tool for Multi-formalism and Meta-Modeling

Even our logos are modeled!


Attributes:
- T :: String
Constraints:
> T


Attributes:
- T :: String
Actions:
> T

Current/Future Work

- implement in AToM³
- investigate limits of approach (∼ boundedness)
- investigate other target formalisms
Questions ?