Patterns for Modeling Inference Rules in a PIM

Kirk Wilson, Ph.D.
Computer Associates International
kirk.wilson@ca.com

Ian Maung, Ph.D.
Computer Associates International
ian.maung@ca.com
Focus on Inference Rules
- If/Then Derivations
- First-order rules
  • Typically implemented in inferencing systems with the Rete algorithm

Modeling first-order rules
- Design patterns and stereotypes for PIM
- Mapping to PSM
Modeling Rules within the PIM

- **Computational semantics of rules**
- **Abstraction**
  - Use parameterized attributes
    - Models rule-based mapping of conditions to results.
    - Stereotype: <<att>> of operation
      - Following Cheesman & Daniels
  - Rules definable through refinement
Inference Rules: The Challenge

- If/Then rules
  - “If” = premise; “Then” = action

- The challenge
  - “Business rules are constraints”
    - But OCL does not allow side effects
  - Keep formal spec purely declarative
    - Avoid specifying algorithmic details
    - Actions expressed at implementation level
UML Patterns

- Class structure
- Safety condition expressed in OCL
  - Modeled as Invariant
- Liveness condition expressed in OCL
  - Guarantees that a change of state will occur
    • Postcondition of an action
Features of Pattern Matching

- Like SQL: PM joins instances of different classes on specified criteria
  - “Match” and “Join” conditions
- PM loops through tuples of matched instances, calling an action on each tuple
- PM allows loops to interact in complex ways without explicit programming
1. Waiting for a Task
   entry/code
   taskCanBeAssigned

2. Waiting for a Res
   entry/code
   resIsAvailable

3. Assign Task to Res
   entry/code
   resAssignedToTask

- PM semantics coded in statechart
  - Mixed-in with process

See Mellor & Balcer, Executable UML, p. 220.
Pattern Matching Structure

Rule: An available Resource is assigned to a Task if it is suitable for the Task.

Inference rule: For all Tasks and Resources, if Task and Resources are unassigned and the Resource is suitable for the Task, then assign the Resource to the Task.
Safety Condition

\[
\text{suitableFor}(\text{task}, \text{resource}) : \text{Boolean}
\]

\[
\text{Inv: Assigner :: assignments->forAll(suitableFor(task, resource))}
\]
Assigner

```
<<att>>
suitableFor(task, resource): Boolean

<<async>>
assignResourceToTask()
```

**guarantee:** unassigned->forAll (assignables->isEmpty)

Model PM as an asynchronous method.

Model PM as an asynchronous method.

**Inv. Assigner::**

\[\text{unassigned} = \text{tasks}->\text{select}(t | t.\text{assignedRes}->\text{isEmpty})\]

-- subset of tasks (not shown)

**Inv. Assigner::**

\[\text{unassignedTo} = \text{resources}->\text{select}(r | r.\text{assignedTo}->\text{isEmpty})\]

-- subset of resources (not shown)

**Inv. Task::**

\[\text{assignables} = \text{unassignedTo}->\text{select}(r | \text{suitableFor}(\text{self}, r))\]

-- Resource: Resources that are suitable for a task.

**Guarantee condition insures liveness.**

- **Assigner::**
  - \[\text{assignment} \rightarrow 0..*\]
  - \[\text{Task} \rightarrow 0,1\]
  - \[\text{assignedTo} \rightarrow 0,1\]
  - \[\text{assignables} \rightarrow 0..*\]
  - \[\text{Resource} \rightarrow 0..*\]

- **Task::**
  - \[\text{assignables} \rightarrow 0..*\]
  - \[\text{Resource} \rightarrow 0..*\]
### Inference Rule Stereotypes

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Stereotype of</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;att&gt;&gt;</td>
<td>Operation</td>
<td>Parameterized attribute</td>
</tr>
<tr>
<td>&lt;&lt;async&gt;&gt; with constraints: guarantee asynchronous</td>
<td>Operation / Method</td>
<td>Operation is to be invoked asynchronously.</td>
</tr>
<tr>
<td>&lt;&lt;guarantee&gt;&gt;</td>
<td>Constraint</td>
<td>Condition must remain true during execution of an operation.</td>
</tr>
</tbody>
</table>
Mapping to the PSM

- Complex in procedural languages
  - Esp. interaction provided by asynchrony

- Inferencing provides:
  - Inferencing conceals implementation semantics from programmer
  - Direct mapping from spec to implementation
    - Asynchrony
    - Guarantee
Derive Specification of the PM Rule’s Action

assign(t:Task, r:Resource)
   pre: t.assignedRes->isEmpty and r.assignedTo->isEmpty and suitableFor(t, r)
       -- Preserves safety
   post: t.assignedRes = r and r.assignedTo = t
       -- Achieves liveness

- Precondition: formally derivable as the *weakest precondition* by which the action can ensure that safety is preserved
- Postcondition: condition that ensures liveness guarantee is met
Implementation of PM Rule

- Modeled by the assignResourceToTask method:
  - Becomes the PM rule whose premise is the universal generalization of the precondition of the action

\[(\forall x)(\forall y) \text{ [IF x is a Task and y is a Resource and } x.\text{assignedRes} = \text{NULL and } y.\text{assignedTo} = \text{NULL and suitableFor}(x, y) \text{ THEN assign}(x, y)]\]
Summary

- It is possible to model semantics of inference rules in a PIM with proper structures and constraints
  - Precise spec of inference rules while omitting irrelevant details
  - Requires safety and liveness conditions
- Profiles for modeling inference rules require minimal extensions to UML
- Inferencing is a PSM consideration
References

Business Rules Community: http://www.BRCommunity.com
Business Rules Working Group of the OMG: stan@hendryxassoc.com