Reaction RuleML for Accessing Loosely-Coupled and Event-Messaged Rule KBs

Adrian Paschke\textsuperscript{1}, Harold Boley\textsuperscript{2}, Paul Vincent\textsuperscript{3}, Davide Sottara\textsuperscript{4}, Tara Athan\textsuperscript{5}

\textsuperscript{1}Freie Universitaet Berlin, Germany
\textsuperscript{2}University of New Brunswick and National Research Council Canada
\textsuperscript{3}Tibco Software Inc., UK
\textsuperscript{4}Arizona State Univ., AZ, USA
\textsuperscript{5}Athan Services, W Lafayette, IN, USA

OMG Technical Meeting, Ontology PSIG, Cambridge, MA, 21 June 2012
Coupling Approaches for (Distributed Rule) Knowledge Bases

- **Strong coupling**
  - Interaction through a stable interface
  - API call is hard coded

- **Loose coupling**
  - Resilient relationship between two or more systems or organizations with some kind of exchange relationship
  - Each end of the transaction makes its requirements explicit, e.g. as an interface description, and makes few assumptions about the other end

- **Decoupled**
  - decoupled in time using (event) messages (e.g. via Message-oriented Middleware (MoM))
  - Often asynchronous stateless communication (e.g. publish-subscribe or CEP event detection)
Reaction Rules: Four Sub-branches

• **Production RuleML**: Production Rules (Condition-Action rules)
• **ECA RuleML**: Event-Condition-Action (ECA) rules
• **CEP RuleML**: Rule-based Complex Event Processing (complex event processing reaction rules, (distributed) event messaging reaction rules, query reaction rules, etc.)
• **KR Reaction RuleML**: Knowledge Representation Event/Action/Situation Transition/Process Logics and Calculi
Info, Life Cycle Mgt.

Interface

Implementation

<Rule @key @keyref @style>

<oid> <!-- object id of the rule --></oid>

<meta> <!-- (semantic) metadata of the rule --></meta>

<scope> <!-- scope of the rule e.g. a rule module --></scope>

<evaluation> <!-- intended semantics --></evaluation>

<signature> <!-- rule signature --></signature>

<qualification> <!-- e.g. qualifying rule declarations, e.g. priorities, validity, strategy --></qualification>

<quantification> <!-- quantifying rule declarations, e.g. variable bindings --></quantification>

<on> <!-- event part --></on>

<if> <!-- condition part --></if>

<then> <!-- (logical) conclusion part --></then>

<do> <!-- action part --></do>

<after> <!-- postcondition part after action, e.g. to check effects of execution --></after>

<else> <!-- (logical) else conclusion --></else>

<elsedo> <!-- alternative/else action, e.g. for default handling --></elsedo>

</Rule>
Reaction RuleML – Example Rule Types

- **Production Rule:**
  ```xml
  <Rule>
    <if>...</if>
    <do>...</do>
  </Rule>
  ```

- **Trigger Rule:**
  ```xml
  <Rule>
    <on>...</on>
    <do>...</do>
  </Rule>
  ```

- **ECA Rule:**
  ```xml
  <Rule>
    <on>...</on>
    <if>...</if>
    <do>...</do>
  </Rule>
  ```
Rule Interface and Rule Implementation

```
<Rule>
  <evaluation>
    <Profile iri="ruleml;definiteProductionRule" direction="forward" style="active"/>
  </evaluation>
  <signature>
    <Atom>
      <op><Rel>likes</Rel></op>
      <arg><Var mode="+"/></arg> <!-- mode=+ i.e. input argument -->
      <arg><Var mode="?"/></arg> <!-- mode=- i.e. input or output argument -->
    </Atom>
  </signature>
  <if>
    <Atom>
      <op><Rel>likes</Rel></op>
      <arg><Var mode="#"/></arg> <!-- mode=+ i.e. input argument -->
      <arg><Ind>wine</Ind></arg>
    </Atom>
  </if>
  <do>
    <Assert>
      <formula>
        <Atom>
          <op><Rel>likes</Rel></op>
          <arg><Ind>John</Ind></arg>
          <arg><Var mode="#"/></arg> <!-- mode=- i.e. input or output argument -->
        </Atom>
      </formula>
    </Assert>
  </do>
</Rule>
```
Separation of Interface and Implementation

Interface 1 (@keyref=r1)

<table>
<thead>
<tr>
<th>Rule keyref=&quot;r1&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;evaluation&gt;&lt;Profile&gt; ...p1... &lt;/Profile&gt;&lt;evaluation&gt;</td>
</tr>
<tr>
<td>&lt;evaluation&gt;&lt;Profile&gt; ...p2 ...&lt;/Profile&gt;&lt;/evaluation&gt;</td>
</tr>
<tr>
<td>&lt;declaration&gt;...s1...&lt;/declaration&gt;</td>
</tr>
<tr>
<td>&lt;/Rule&gt;</td>
</tr>
</tbody>
</table>

Interface 2 (@keyref=r2)

<table>
<thead>
<tr>
<th>Rule keyref=&quot;r2&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;evaluation&gt;&lt;Profile&gt; ...p3... &lt;/Profile&gt;&lt;evaluation&gt;</td>
</tr>
<tr>
<td>&lt;declaration&gt;...s2...&lt;/declaration&gt;</td>
</tr>
<tr>
<td>&lt;/Rule&gt;</td>
</tr>
</tbody>
</table>

Implementation 2 (@key= r2)

<table>
<thead>
<tr>
<th>Rule key=&quot;r2&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;if&gt; ... &lt;/if&gt;</td>
</tr>
<tr>
<td>&lt;do&gt; &lt;/do&gt;</td>
</tr>
<tr>
<td>&lt;/Rule&gt;</td>
</tr>
</tbody>
</table>

Implementation 1 (@key=r1)

<table>
<thead>
<tr>
<th>Rule key=&quot;r1&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;if&gt; ... &lt;/if&gt;</td>
</tr>
<tr>
<td>&lt;do&gt; &lt;/do&gt;</td>
</tr>
<tr>
<td>&lt;/Rule&gt;</td>
</tr>
</tbody>
</table>
Messaging Reaction Rules

<Rule>
...
<do><Send> query </Send></do>
<on><Receive> response </Receive></on>
<if> prove some conditions, e.g. make decisions on the received answers </if>
....
</Rule>

Note: The „on“, „do“, „if“ parts can be in arbitrary combinations to allow for a flexible workflow-style logic with subconversations and parallel branching logic.
Send and Receive Messaging

- Send a message
  - Send (@directive oid, protocol, agent, content)

- Receive a message
  - Receive (@directive oid, protocol, agent, content)

- oid is the conversation identifier (enabling also subconversations)
- protocol: protocol definition (high-level protocols and transport prot.)
- agent: denotes the target or sender of the message
- @directive: pragmatic context, e.g. FIPA Agent Communication Language (ACL) primitives
- content: Message payload
Loosley-Coupled Communication via Messages to Agent Interface

Event Message is local to the conversation state (oid) and pragmatic context (directive)
Decoupled Event Messaging
Reaction Rules

Complex Event Algebra Operators are used to define CEP patterns

Note: The event messages are decoupled from the sender (in contrast to the loosely coupled send and receive communication using rule (agent) interfaces.
Reaction RuleML Metamodel

Top-Level Ontologies

General concepts such as space, time, event, action and their properties and relations

Spatio Ontology
Temporal Ontology
Action Ontology
Event Ontology
Situation Ontology
Process Ontology
Agent Ontology

Domain Ontologies

Task Activities Ontologies

Application Ontologies

Vocabularies related to specific domains by specializing the concepts introduced in the top-level ontology

Vocabularies related to generic tasks or activities by specializing the concepts introduced in the top-level ontology

Specific user/application ontologies

E.g. ontologies describing roles played by domain entities while performing application activities
Example: Situation Top Ontology Model

- **Situation Properties**
  - (time, location, participants, ...)

- **Situation Content**

- **Heterogeneous Situation**
  - Dynamic Change Situation
  - Time Frame Situation
  - Frequency Situation

- **Homogenous Situation**
  - State Situation
  - Process Situation
  - Iterative Situation

- **Iterative Situation**
  - He Runs
  - He Coughs
  - He Smokes

- **Habitual Situation**
  - TrafficLight Changes
  - Within 5 Minutes
  - Rings 3 Times
  - Lays On The Floor

Use the other top ontologies to describe situations.

**Situation Descriptions**

**Situation Types**

**Situation Individuals**
Complex Event / Action Algebra Operators defined in Metamodel

• Action Algebra
  *Succession* (Ordered Succession of Actions), *Choice* (Non-Deterministic Choice), *Flow* (Parallel Flow), *Loop* (Loops), *Operator* (generic Operator which can point to an external Action metamodel/ontology)

• Event Algebra
  *Sequence* (Ordered), *Disjunction* (Or), *Xor* (Mutual Exclusive), *Conjunction* (And), *Concurrent*, *Not*, *Any*, *Aperiodic*, *Periodic*, *AtLeast*, *AtMost*, *Operator* (generic Operator which can point to an external event metamodel/ontology)

• Interval Algebra (Time/Spatio/Event/Action/… Intervals)
  *During*, *Overlaps*, *Starts*, *Precedes*, *Meets*, *Equals*, *Finishes*, *Operator* (generic Operator which can point to an external Interval metamodel/ontology)
Reaction RuleML Metamodel and External Ontologies + Data - Examples

<Quantifier type="ruleml:Forall"> == <Forall>
<Operator type="ruleml:And"> == <And>
<Operator type="ruleml:Conjunction"> == <Conjunction>
<Negation type="ruleml:InflationaryNegation"> == <Naf>
<Action type="ruleml:Assert"> == <Assert>
<Action type="ruleml:Retract"> == <Retract>
<Event type="ruleml:SimpleEvent"> == <Atom> ... </Atom>
<Event type="ibm:CommonBaseEvent"> == IBM CBE
<Operator type="snoop:Squence"> == Snoop Algebra
   == <Operator type="ruleml:Sequence"> == <Sequence>
<Ind iri="person.xml#xpointer(/Person/LastName[1]/text())"/>
<Action iri="BPEL.xml#xpointer(/invoke[@name=checkHotel])">
Execution Semantics
(defined in the evaluation semantics element)

1. Definition
   – Definition of event/action pattern e.g. by event algebra
   – Based on declarative formalization or procedural implementation
   – Defined over an atomic instant or an interval of time, events/actions, situation, transition etc.

2. Selection
   – Defines selection function to select one event from several occurred events
     (stored in an event instance sequence e.g. in memory, database/KB) of a particular type,
     e.g. “first”, “last”
   – Crucial for the outcome of a reaction rule, since the events may contain different
     (context) information, e.g. different message payloads or sensing information

3. Consumption
   – Defines which events are consumed after the detection of a complex event
   – An event may contribute to the detection of several complex events, if it is not consumed
   – Distinction in event messaging between “multiple receive” and “single receive”
   – Events which can no longer contribute, e.g. are outdated, should be removed

4. Execution
   – Actions might have an internal effect i.e. change the knowledge state leading to state
     transition from (pre-)condition state to post-condition state
   – The effect might be hypothetical (e.g. a hypothetical state via a computation) or
     persistent (update of the knowledge base),
   – Actions might have an external side effect
Summary of Selected Reaction RuleML Features

• Different Rule Families and Types
• Rule Interface and Implementation
• Decoupled event messaging and loosely-coupled send/receive interaction against rule (KB) interface within conversations, coordination/negotiation protocols and pragmatic directives
• External data models and ontologies / metamodels, e.g.
  \textit{time, spatio, event, action, situation, process, agent}
• Different detection, selection and consumption semantics
• Action Algebra, e.g.
  \textit{Succession (Ordered Sequence), Choice (Non-Deterministic Selection), Flow (Parallel Concurrent Flow), Loop (Iteration), Operator (generic Operator)}
• Event Algebra, e.g.
  \textit{Sequence (Ordered), Disjunction (Or), Xor (Mutal Exclusion), Conjunction (And), Concurrent, Not, Any, Aperiodic, Periodic, Operator (generic)}
• Intervals (Time, Event)
• Situations (States, Fluents)
• External event query languages
• ...
Modular Relax NG for Reaction RuleML

• Reaction RuleML is a monotonic language extension of Deliberation RuleML
  – Every valid Deliberation RuleML document is a valid Reaction RuleML document

• Deliberation RuleML 1.0 is specified by a modular, monotonic Relax NG schema, released 2012-04-03
  – Reaction RuleML 1.0 can be specified by new Relax NG modules that may be freely-combined with (some or all) Deliberation RuleML modules
  – E.g.: modules/Rule.rnc, modules/scope.rnc, modules/event.rnc, modules/action.rnc, ...

• MYNG (http://ruleml.org/1.0/myng) provides a GUI for building a customized Deliberation RuleML Relax NG schema from these modules
  – MYNG can be extended to the Reaction RuleML schema modules
Rule interchange (on the PIM level)
Relations between Semantic OMG Standards

Ontology Definition Metamodel (ODM)

Semantics for Business Vocabularies & Rules (SBVR)

Business Rules (CIM)

RuleML Interchange, Interaction, and Transformation (PIM)

Production Rule Representation (PRR)

Decision Model Notation (DMN)

Modelling of Rules (PIM)

Direct Mapping for OWL Formal Grounding (CL)

Vocabularies in ODM and other Ontology languages via ODM (PIM)

Mappings for e.g. planned Event Metamodel Profile (EMP), Agent Metamodel and Profile, ... (specific semantic models)

Mappings for ER, Logical DB, XML Schema, ... (Data)

Information Management Metamodel (IMM)
Example: Semantic BPM Approach

1. OMG BPMN (+ Semantics) (CIM)

2. BPEL (PIM)

3. RuleML Rule Responder (PIM)

4. Prova Rule Engine (PSM)

% receive query and delegate it to another party
rcvMsg(CID,esb, Requester, acl_query-ref, Query) :
  responsibleRole(Agent, Query),
  sendMsg(Sub-CID,esb,Agent,acl_query-ref, Query),
rcvMsg(Sub-CID,esb,Agent,acl_inform-ref, Answer),
... (other goals)...
sendMsg(CID,esb,Requester,acl_inform-ref,Answer)
Example: Semantic BPM
Rule Responder Project
Example: Hybrid Knowledge Bases in Drools
Example: Semantic Event Processing in Government

1. Map ontologies to s/w devt env
2. Execute against known channels
3. Analyze data for new / revised classifications, data criteria etc
4. Use models to revise ontology + rule parameters
5. Define & run self-analysis rules

For example see http://semanticommunity.info/@api/deki/files/18219/BrandNieman06182012.pptx
Thank you!

Questions?

More information: