

# On Interchange between JBossRules and Jess

Oana Nicolae<sup>1</sup>   Adrian Giurca<sup>1</sup>   Gerd Wagner<sup>1</sup>

<sup>1</sup>Department of Internet Technology  
Institute of Informatics  
Brandenburg Technical University at Cottbus, Germany

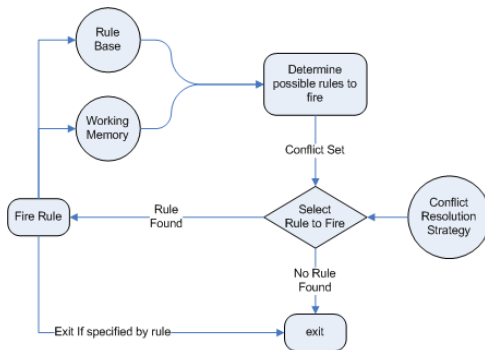


1st International Symposium on Intelligent and Distributed  
Computing, 2007  
Craiova, Romania

## Actual Business Rules Context

- There are different rule-based language implementations:
  - Object Oriented Rule Languages: **JBossRules**, JRules, Fair Isaac, Oracle Business Rules
  - Artificial Intelligence Rule Languages: **Jess**, Prolog
  - Semantic Web Rule Languages: JenaRules, SWRL
- Rule-based vendor tools are seriously competed by open-source rule engines/platforms.
- Inter-operability initiatives are towards a commonly agreed Rule Interchange Format.

## Forward-chaining engines



**Figure:** Forward-Chaining Rule Engine (Drools).

## JBossRules aka Drools

- Rule engine based on an enhanced version of Rete algorithm.
- Open-source, Object-Oriented Production Rules System written entirely in Java language.
- Separation of the Business Logic (the rules) from Business Data (the facts).
- Light and easy to understand syntax (i.e. DRL syntax) uses Java to express field constraints, functions and consequences.
- Easy to integrate with the mainstream JEE5 technologies.
- Employs complex rules and works with large data sets.

### Drools Framework

<http://www.jboss.com/products/rules>  
<http://labs.jboss.com/drools/downloads.html>

## JBoss Rule Example

*If the driver is male and is under the age of 25, then the driver is young driver.*

```
rule "Young-Driver"
  agenda-group "driver eligibility"
  salience 10
  when
    $driver:Driver(gender == "male", age < 25)
  then
    $driver.setDriverAgeCategory("young driver");
    modify($driver);
  end
```

Rule name

attributes

Conditions or LHS (when)

Actions or RHS (then)

*Logically means:*

```
if
  driver($driver) AND
  gender($driver, "male") AND
  age($driver, 25)
then
  assert(driverAgeCategory($driver, "young driver"))
```

# Jess

- The most popular AI language.
- Employs a Rete-based rule engine and scripting environment for the Java platform that gives access to all of Java's APIs.
- Provides forward-chaining, backward-chaining and working memory queries.
- The language syntax is very expressive and can express complex logical relationships with very little code.
- Is a highly specialised form of Lisp, as the main unit of language syntax is the list.
- Offers sophisticated forms to encode the business data (i.e. the facts):  
unordered facts, ordered facts and shadow facts.
- Can be licensed for commercial use and is available at no cost for academic use.

## Jess

```
http://herzberg.ca.sandia.gov/jess/  
http://www.jessrules.com/jess/download.shtml
```

## Jess 7.0 - Simplified Syntax

- Adds a simplified language syntax with new-style slot descriptions enclosed in curly braces.
- Slot descriptions (aka Java Beans properties) are Java-like boolean expressions constructed using Java valid operators in infix format.
- Within this simplified pattern, for each mentioned slot in LHS, a variable named after the slot is automatically created and is available on the RHS of the rule.

```
(defrule Young-Driver
  ?driver <- (Driver{gender == "male" &&
                age < 25})
=>
  (modify ?driver (driverAgeCategory "young driver"))
  (printout t "The driver is " ?gender "and has"
            ?age "years old." clrf)
)
```

Diagram annotations:

- Rule name: (defrule Young-Driver
- Conditions or LHS (when): ?driver <- (Driver{gender == "male" && age < 25})
- Actions or RHS (then): (modify ?driver (driverAgeCategory "young driver")) (printout t "The driver is " ?gender "and has" ?age "years old." clrf)

## Initiatives on Rule Interoperability

- Initiatives on rules inter-operability have been started from both UML modelers and ontology architects communities.
- They emphasise the important role of business rules inter-operability initiatives, so that rule-system developers can do their work without concern about a vendor-specific format and in particular without concern about the compatibility between the technologies.
- Our work has followed the principles initiated on rules inter-operability such as the **RuleML**<sup>1</sup>, **OMG Production Rules Representation**<sup>2</sup>, **W3C**<sup>3</sup> (i.e. RIF - Rule Interchange Format) and **EU Network of Excellence REVERSE**<sup>4</sup> (i.e. R2ML - REVERSE I1 Rule Markup Language).

---

<sup>1</sup>RuleML - <http://www.ruleml.org>

<sup>2</sup>OMG PRR - <http://www.omg.org/docs/bmi/07-03-05.pdf>

<sup>3</sup>RIF - <http://www.w3.org/2005/rules/>

<sup>4</sup>R2ML - <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=R2ML>

## Rule Interchange Markup Language

- The **REVERSE I1 Rules Framework** developed Rule Interchange between different rule platforms via R2ML. The interchange is performed with the help of **R2ML translators**<sup>5</sup>.
- We use **R2ML 0.4** as Interchange Language Format. It is a mature and experienced enough rule interchange language to provide a concrete interchange format for different rule systems and languages.
- Our rule interchange work addresses **JBossRules**<sup>6</sup> 3.0.6 as source platform and **Jess**<sup>7</sup> 7.0 as a target platform, bridging this way an Object-Oriented rule language to an Artificial Intelligence rule language.

---

<sup>5</sup>Translators - <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=node/15>

<sup>6</sup>JBossRules - <http://labs.jboss.com/jbossrules/>

<sup>7</sup>Jess - <http://herzberg.ca.sandia.gov/jess/>

# Mapping of Conditions I

## JBossRules Implementation

```
$driver:Driver(gender == "male")
```

## R2ML Serialisation

```
<r2ml:conditions>  
<r2ml:AttributionAtom r2ml:attributeID="userv:Driver.gender">  
  <r2ml:subject>  
    <r2ml:ObjectVariable r2ml:name="driver" r2ml:classID="userv:Driver"/>  
  </r2ml:subject>  
<r2ml:dataValue>  
  <r2ml:TypedLiteral r2ml:lexicalValue="male" r2ml:datatypeID="xs:string"/>  
</r2ml:dataValue>  
</r2ml:AttributionAtom>  
</r2ml:conditions>
```

## Jess Implementation

```
?driver <-(Driver{gender == "male"})
```

## Mapping of Conditions II

### JBossRules Implementation

```
$driver:Driver (age < 25)
```

### R2ML Serialisation

```
<r2ml:conditions>  
<r2ml:DatatypePredicateAtom r2ml:datatypePredicateID="swrlb:lessThan">  
  <r2ml:dataArguments>  
    <r2ml:AttributeFunctionTerm r2ml:attributeID="user:Driver.age">  
      <r2ml:contextArgument>  
        <r2ml:ObjectVariable r2ml:name="driver" r2ml:classID="user:Driver"/>  
      </r2ml:contextArgument>  
    </r2ml:AttributeFunctionTerm>  
    <r2ml:TypedLiteral r2ml:datatypeID="xs:integer" r2ml:lexicalValue="25"/>  
  </r2ml:dataArguments>  
</r2ml:DatatypePredicateAtom>  
</r2ml:conditions>
```

### Jess Implementation

```
?driver <-(Driver{age < 25})
```

## Mapping of Actions

### JBossRules Implementation

```
$driver.setDriverAgeCategory("young driver");  
modify($driver);
```

### R2ML Serialisation

```
<r2ml:producedAction>  
<r2ml:AssignActionExpression r2ml:propertyID="userv:Driver.driverAgeCategory">  
  <r2ml:contextArgument>  
    <r2ml:ObjectVariable r2ml:name="driver" r2ml:classID="userv:Driver"/>  
  </r2ml:contextArgument>  
  <r2ml:TypedLiteral r2ml:lexicalValue="young driver"  
    r2ml:datatypeID="xs:string"/>  
</r2ml:AssignActionExpression>  
</r2ml:producedAction>
```

### Jess Implementation

```
(modify ?driver (driverAgeCategory "young driver"))
```

## About Soundness

- Neither JBossRules nor Jess don't provide a semantics of their languages. The soundness of the proposed translation was established by testing rules.
- We translate the JBoss production rules into Jess implementation via R2ML and execute those rules based on analogous facts from the Working Memory.
- The *soundness* of the translation is concerned with obtaining the same results (i.e. the same facts in the Working Memory).

## ... and Completeness

- The translation is *incomplete* i.e. not all possible Drools rules can be translated into corresponding Jess rules, mainly because of the unstructured action part in Drools rules.
- Translation from PSM (JBossRules) to PIM (R2ML) implies the access to the rules vocabulary (Java beans classes).
- JBossRules PR actions find their mapping in the R2ML PR Actions. Since JBossRules allows free Java code in the rules action part it cannot be completely mapped to R2ML actions.
- One limitation of the R2ML to Jess syntax translation is that it can only be used with unordered facts.

## Conclusions and Future Work

- The paper provides a description of rule translation from JBoss Rules, an Object Oriented rule language, into Jess, an Artificial Intelligence rule language using R2ML as interchange format.
- Our future work intends to finalise the implementation of all transformation proposed in the paper.
- Intend to stay close to the Drools idea to incorporate native code of pluggable dialects (e.g. Jess) inside Drools rules.

## Our work includes

- R2ML

<http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=R2ML>

- Interchange Web Service

<http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=node/27>

- Strelka

<http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=Strelka>

- Translators

<http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=node/15>