Using Constraint Solvers as Inference Engines to Validate and Execute Rules-Based Decision Models

Jacob Feldman, Ph.D.
Founder & CTO, OpenRules, Inc.

JSR-331 Specification Lead
Two Types of Rule Engines

• **Inferential** Rule Engines
  – support a pure declarative representation of business rules
  – Rete-based

• **Sequential** Rule Engines
  – rely on user-defined sequencing of rules and rule families
Rule Engines

- Data
- Rules in Rule Language
- Working Memory
  - Match Rules
  - Resolve Conflicts
  - Fire Rules
- Results

Focal Point: Rete-based Implementation

~ 1982-1995
BRMS

Focal Point: Rules Organization

Data

Inference Rule Engine

Results

Business Rules Management Interface

Rules Repository

Rules in Rule Language

Sequential Rule Engine

~ 1996 - Current

© 2011 OpenRules, Inc.
Inference Engines: Rete Domination

• It works!
• A great algorithm! Keeps improving...
• A good platform for other systems (e.g. JBoss CEP, Planner)
• Why no alternatives?
  – The best within its framework:
  – Multiple commercial and open source implementations
  – No needs for alternative
  – Inference frequently is not needed
From BRMS to BDMS: Decision Management Systems

• The newest Decision Modeling approach specifies such an organization of decisions and supporting rules that allows us to completely automate their execution (without coding!)

• Real orientation to Business Users

• Current Decision Modeling Efforts:
  – OMG DMN (Decision Modeling Notation)
  – The Decision Model from KPI

© 2011 OpenRules, Inc.
## Rule Family Examples

### RuleFamily PersonLikelihoodOfDefaultingOnLoan

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
<th>Condition</th>
<th>Condition</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person Employment History</td>
<td>Person Mortgage Situation</td>
<td>Person Miscellaneous Loans Assessment</td>
<td>Person Outside Credit Score</td>
<td>Person Likelihood of Defaulting on a Loan</td>
</tr>
<tr>
<td>Is</td>
<td>Poor</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
</tbody>
</table>

### RuleFamily PersonEmploymentHistory

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person Years at Current Employer</td>
<td>Person Number of Jobs in Past Five Years</td>
<td>Person Employment History</td>
</tr>
<tr>
<td>&lt; 1</td>
<td>&gt; 5</td>
<td>Is Poor</td>
</tr>
<tr>
<td>&lt; 1</td>
<td>&lt;= 5</td>
<td>Is Average</td>
</tr>
<tr>
<td>Within [1;2]</td>
<td>&gt; 5</td>
<td>Is Poor</td>
</tr>
<tr>
<td>Within [1;2]</td>
<td>&lt;= 5</td>
<td>Is Average</td>
</tr>
<tr>
<td>&gt; 2</td>
<td>&lt;= 4</td>
<td>Is Good</td>
</tr>
<tr>
<td>&gt; 2</td>
<td>Within (4;6)</td>
<td>Is Average</td>
</tr>
<tr>
<td>&gt; 2</td>
<td>&gt; 6</td>
<td>Is Poor</td>
</tr>
</tbody>
</table>
Rules: To Order or Not to Order

• The Decision Model promotes two important principles:
  – The order of Rules inside Rule Families should not matter. In particular, rule overrides are not allowed
  – Rule Families (even when they are inferentially linked) can be defined in any order
• Obviously, sequential rule engines cannot be used to satisfy these principles
Implementing Decision Models

• Only Rule Engines with true inference capabilities can support these principles
• It brings a new incentive to implement inference engines that are capable to execute decision models
• Question: Is it necessary to convert a decision model to a rule language that can be executed by a Rete-based engine?
Two Ways of Executing Decision Models

1. Converter to a Rule Language
   - Rete-based Rule Engine
   - Decision Model
   - Data
   - The Same Results

2. Generator of an intermediate Constraint Satisfaction Problem
   - Constraint-based Rule Engine
   - Decision Model
   - Data
   - Business Users

Business Decision Management Interface

© 2011 OpenRules, Inc.
Constraint-based Rule Engine

• Constraint-based Programming (CP) and Rules-based Programming (BR) are similar declarative technologies that deal with similar decision support problems.

• However, until recently their input was different:
  – Business Rules were expressed using different flavors of a Rule Language oriented to Rete Engines.
  – Constraint Satisfaction Problems (CSP) were expressed using different flavors of a Constraint Language oriented to Constraint Solvers.

• These days CP and BR become standardized:
  – The Decision Model as a common input for CP and BR engines.
  – The Decision Model becomes a de-facto standard for BDMS (?
  – JSR-331 provides a standard API for different Java-based CP solvers.
The Decision Model

• The Decision Model (TDM) is oriented to business users allowing them to describe their decisions and supporting rule families without help from developers

• TDM authors claim that today ~50% of the US financial institutions adopted or consider to adopt TDM

• TDM does not use any rule language

• TDM specifies strict rules organization principles

• Current implementations:
  – Sapiens, OpenRules, Corticon, New Wisdom, ...
JSR-331 – Java Specification Request

• JSR 331 is “Constraint Programming API” standard developed within the Java Community Process (JCP) [www.jcp.org](http://www.jcp.org)

• JSR-331 covers key concepts and design decisions related to the standard representation and resolution of constraint satisfaction and optimization problems

• Reached The Final Draft phase

• Currently has 3 working implementations
OpenRules Approach

• Since its inception in 2003, OpenRules supported two engines:
  – Sequential Rule Engine
    • for efficient execution of hierarchies of complex decision tables defined in Excel files (with Java snippets)
  – Rule Solver
    • A constraint-based engine that was used when real inference and/or optimization were required
    • However, an input for Rule Solver used to be created by a user familiar with Constraint Programming concepts
News from OpenRules

• A new Rule Solver can execute inferential decision models “as is”!
• No additional coding required
  – A user do not have to learn neither a rule language or a constraint language
• A new Rule Solver can handle real inference (similarly to Rete)
Actions are better than words: 
Rule Solver Demo

• Simple Decision Model “Hello Customer”
• No order between Rule Families
• Conflicts inside Rule Families
"Good Afternoon, Mrs. Robinson!"
# Rule Families

## RuleFamily DefineGreeting

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
<th>Conclusion</th>
<th>Greeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Hour</td>
<td>Current Hour</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$\leq 11$</td>
<td>Is</td>
<td>Good Morning</td>
</tr>
</tbody>
</table>

## RuleFamily DefineAgeGroup

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
<th>Conclusion</th>
<th>Age Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>Age</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$\leq 7$</td>
<td>Is</td>
<td>Little</td>
</tr>
</tbody>
</table>

## RuleFamily DefineSalutation

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
<th>Condition</th>
<th>Age Group</th>
<th>Conclusion</th>
<th>Salutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td>Marital Status</td>
<td>Age Group</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is Female</td>
<td>Is Married</td>
<td>Is</td>
<td>Little</td>
<td>Is</td>
<td>Little</td>
</tr>
<tr>
<td>Is Female</td>
<td>Is Single</td>
<td>Is Not</td>
<td>Little</td>
<td>Is</td>
<td>Mrs.</td>
</tr>
<tr>
<td>Is Male</td>
<td></td>
<td>Is</td>
<td>Mr.</td>
<td></td>
<td>Ms.</td>
</tr>
</tbody>
</table>
# Defining Business Glossary

<table>
<thead>
<tr>
<th>Fact Name</th>
<th>Business Concept</th>
<th>Attribute</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td></td>
<td>gender</td>
<td>Male, Female</td>
</tr>
<tr>
<td>Marital Status</td>
<td></td>
<td>maritalStatus</td>
<td>Single, Married</td>
</tr>
<tr>
<td>Age</td>
<td>Customer</td>
<td>age</td>
<td>1-120</td>
</tr>
<tr>
<td>Age Group</td>
<td></td>
<td>ageGroup</td>
<td>Little, Young, Adult</td>
</tr>
<tr>
<td>Current Hour</td>
<td></td>
<td>hour</td>
<td>0-24</td>
</tr>
<tr>
<td>Greeting</td>
<td>Response</td>
<td>greeting</td>
<td>Good Morning, Good Afternoon, Good Evening, Good Night</td>
</tr>
<tr>
<td>Salutation</td>
<td></td>
<td>salutation</td>
<td>Mr., Ms., Mrs., Little</td>
</tr>
</tbody>
</table>
Defining Test Data (in Excel)

<table>
<thead>
<tr>
<th>Customer Name</th>
<th>maritalStatus</th>
<th>gender</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robinson</td>
<td>Married</td>
<td>Female</td>
<td>24</td>
</tr>
<tr>
<td>Smith</td>
<td>Single</td>
<td>Male</td>
<td>6</td>
</tr>
</tbody>
</table>
Executing Decision Model

- **Using Sequential Rule Engine**

<table>
<thead>
<tr>
<th>Decision</th>
<th>Rule Description</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>DetermineCustomerGreeting</td>
<td>Define Greeting Word</td>
<td>Greeting Is Good Morning</td>
</tr>
<tr>
<td>DetermineCustomerGreeting</td>
<td>Define Age Group</td>
<td>Age Group Is Little</td>
</tr>
<tr>
<td>DetermineCustomerGreeting</td>
<td>Define Salutation Word</td>
<td>Salutation Is Little</td>
</tr>
</tbody>
</table>

Decision has been finalized
Decision: Good Morning, Little Smith!

- **Using Inferential Rule Solver**

```plaintext
=== Before Posting Rule Family Constraints
Marital Status[Single]
Greeting[Good Morning, Good Afternoon, Good Evening, Good Night]
Age[6]
Age Group[Little, Young, Adult]
Gender[Male]
Salutation[Mr., Ms., Mrs., Little]
Current Hour[0]
=== Summary after execute() ===
Marital Status[Single]
Greeting[Good Morning]
Age[6]
Age Group[Little]
Gender[Male]
Salutation[Little]
Current Hour[0]
Decision: Good Morning, Little Smith!
```
Let’s change the decision order

• Ordered Decisions

<table>
<thead>
<tr>
<th>Decision DetermineCustomerGreeting</th>
<th>Execute Rule Families</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Greeting Word</td>
<td>:= DefineGreeting()</td>
</tr>
<tr>
<td>Define Age Group</td>
<td>:= DefineAgeGroup()</td>
</tr>
<tr>
<td>Define Salutation Word</td>
<td>:= DefineSalutation()</td>
</tr>
</tbody>
</table>

• Unordered Decisions

<table>
<thead>
<tr>
<th>Decision DetermineCustomerGreeting</th>
<th>Execute Rule Families</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define Greeting Word</td>
<td>:= DefineGreeting()</td>
</tr>
<tr>
<td>Define Salutation Word</td>
<td>:= DefineSalutation()</td>
</tr>
<tr>
<td>Define Age Group</td>
<td>:= DefineAgeGroup()</td>
</tr>
</tbody>
</table>

Define Age Group goes after Define Salutation
Executing Decision Model Again

• Using Sequential Rule Engine

```
Decision DetermineCustomerGreeting: Define Greeting Word
Conclusion: Greeting Is Good Morning
Decision DetermineCustomerGreeting: Define Salutation Word
Decision DetermineCustomerGreeting: Define Age Group
Conclusion: Age Group Is Little
Decision has been finalized
Decision: Good Morning, null Smith!
```

• Using Inferential Rule Solver

```
=== Before Posting Rule Family Constraints
Marital Status[Single]
Greeting[Good Morning, Good Afternoon, Good Evening, Good Night]
Age[6]
Age Group[Little, Young, Adult]
Gender[Male]
Salutation[Mr., Ms., Mrs., Little]
Current Hour[0]
=== Summary after execute() ===
Marital Status[Single]
Greeting[Good Morning]
Age[6]
Age Group[Little]
Gender[Male]
Salutation[Little]
Current Hour[0]
Decision: Good Morning, Little Smith!
```

Now Salutation remains undefined (null)

Rule Solver continues to work well!
Let’s create an invalid Rule Family

• Correct Rule Family

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
<th>Conclusion</th>
<th>Salutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td>Marital Status</td>
<td>Age Group</td>
<td></td>
</tr>
<tr>
<td>Is Female</td>
<td>Is Married</td>
<td>Is Not Little</td>
<td>Is Little</td>
</tr>
</tbody>
</table>
| Is Male         | Is Single  | Is Not Young | Is Young

• Incorrect Rule Family

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition</th>
<th>Conclusion</th>
<th>Salutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td>Marital Status</td>
<td>Age Group</td>
<td></td>
</tr>
<tr>
<td>Is Female</td>
<td>Is Married</td>
<td>Is Not Little</td>
<td>Is Little</td>
</tr>
</tbody>
</table>
| Is Male         | Is Single  | Is Not Young | Is Young

• Rule Solver diagnoses the error

* Failure to post constraint: IF Gender = Male AND Age Group != Young THEN Salutation = Mr.

• Unfortunately, a sequential Rule Engine produces “correct” results – a user will not notice a problem!
Rule Solver Validates and Executes Decision Models

1. Convert to a Rule Language
   - Rete-based Rule Engine

2. Generate an intermediate CSP using JSR-331 API
   - Constraint-based Rule Engine

The Same Results
How Rule Solver Works

• Takes an Excel-based Decision Model as an input
• Generates a constraint satisfaction problem (CSP)
  – Done on the fly, no code generation
  – Uses JSR-331 “Constraint Programming API” standard
• Solves the CSP using any Constraint Solver compliant with JSR-331 (there are at least 3 CP solvers available today)
• Produces errors (if any) in business terms of the initial decision model
• Saves the results in the business objects
How Rule Solver Works

Create and Solve a CSP
BR and CP are Similar

<table>
<thead>
<tr>
<th>Business Rules (BR)</th>
<th>Constraint Programming (CP)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Objects</strong> with some yet unknown characteristics</td>
<td><strong>Decision variables</strong> with yet unknown values from a finite domain</td>
</tr>
<tr>
<td><strong>Rules</strong> specify relations among these objects</td>
<td><strong>Constraints</strong> specify relations among these variables</td>
</tr>
<tr>
<td><strong>Rule Engine</strong> executes these rules using Rete algorithm to find unknown characteristics of the business objects</td>
<td><strong>Constraint Solver</strong> uses different search strategies to find an assignment of possible values to variables that satisfies all constraints</td>
</tr>
</tbody>
</table>
BR and CP are Different

• BR Advantage:
  – Rules Repository is managed by business people while Constraint Store is usually under control of software developers

• CP Advantage:
  – Rules usually have to consider All (!) possible combinations of the problem parameters
  – Constraints do not have to cover all situations but rather define an optimization objective and allow a search algorithm to find an optimal solution
  – CP allows to choose different search strategies (while BR relies only on one Rete algorithm)
Miss Manners: Minimizing Rules Violation

- All rules violations may have an attached cost, e.g. 1. As a result, this solution has the minimal total constraint violation cost 8.

- However another solution looks “better”:


JSR-331 demonstrates the proper implementation

Presented at RulesFest-2009
Consistency Validation (1)

- Validate Rules Consistency using Constraint Propagation:
  - Rule Solver simply posts all constraints one-by-one with constraint propagation turned on.
  - If a constraint fails to be posted, a user will be notified that the associated rule is in conflict with the rules, for which the corresponding constraints were previously posted.
Consistency Validation (2)

• Validate Rules Consistency using Test Data:
  – Before posting any constraints, Rule Solver is trying to instantiate constrained variables, for which the proper test data is defined. If an error occurs, the user will be informed about invalid data.
  – If there are no errors in the data, then Rule Solver will try to post all constraints (with data constraints already posted)
    • If a constraint fails to be posted, the user will be notified that the associated rule is in conflict with previously posted constraints (rules).
    • To help a user find the reason for the conflict, Rule Solver displays the current state of all instantiated (or only partially instantiated) variables corresponding to the fact types.
Completeness Checking

• Rules Completeness:
  – Rule Solver determines whether all constrained variables have been instantiated for all conclusion fact types
  – If not, Rule Solver points to the fact types with remaining possible values from their domains. This prompts a user to identify which rules should be extended to cover the remaining situations
  – Rule Solver validates consistency of not only one Rule Family but of all Rule Families included in the Decision Model and related through inferential relationships
OpenRules Decision Models

• Executable Decision Models
  – Can be created and tested by business people using MS Excel or Google Docs
  – A user may decide to use either Sequential or Inference rules execution modes

• Custom Decision Models
  – Easily configurable decision model templates defined in Excel specify:
    1) what is actually allowed;
    2) new custom features
  – Open source
Summary

• The Decision Model provides a common input for Rete-based or Constraint-based rule engines
• Rule Solver can be used instead an real (non-sequential) rule engines
• Rule Solver validates compliance with The Decision Model inferential principles
• Rules Solver may go beyond traditional BR problems

www.OpenRules.com