Rule-based Computing in Industry

—

Concepts, Issues, and Perspectives

Hassan Aït-Kaci

Senior Technical Staff Member

IBM Canada Ltd.

Outline

- **Business Rules**
  - What are they?
  - What are they used for?
  - How are they used?
- **Technical Issues**
  - Object models
  - Static analysis
- **Perspectives**
  - Rule interchange
  - Webization
- **Conclusion**

Business Rules—*What are they?*

The term “Business rule” designates a means to specify a decision-making process using condition/action (C/A) rules

Example of a Business rule:

```
r1: if   o: Order(value >= 10000)
    then o.discount += 20;
```

Such rules are packaged in rulesets organized in process flows

Business Rules—*What are they used for?*

Business rules are used for:

- filtering data
- processing orders
- profiling customers
- automating business processes
- *etc.* ...

Business rules are direct descendants of Expert Systems!

Business Rules—***How are they used?***

- Business rules are specified by business people—not by programmers
- They use a Business Rule Management System (BRMS)
- A BRMS is a high-level interface taking pseudo-natural language input
- They generate C/A rules acting on objects

Technical Issues—*Formalizing business rules*

We need to define formally such informal notions as:

- Object instance
- Object pattern
- Conditional object pattern
- Working memory
- Rule and ruleset
- Pattern matching
- Rule instance
- Rule action
- Rule application
- Rule engine

Formalizing business rules—*Object instance*

- Object instance:

$$\texttt{o:O(}\; T_1\; a_1 = v_1,\; \vdots\; T_n\; a_n = v_n\texttt{)}$$

- *E.g.*,

```
p:Person( int age = 27,
          boolean isVoter = true)
```

Formalizing business rules—*Object pattern*

- Object pattern:

$$\texttt{o:O(}\ T_1\ a_1 = x_1,\ \vdots\ T_n\ a_n = x_n\texttt{)}$$

- *E.g.*,

```
p:Person( int age = a,
          boolean isVoter = b)
```

Formalizing business rules—*Conditional object pattern*

- Conditional object pattern:

$$o{:}O\ \big(\ T_1\ a_1 = x_1, \ldots\ T_n\ a_n = x_n$$
$$\| \ c_1(o, x_1, \ldots, x_n), \ldots, c_m(o, x_1, \ldots, x_n)\big)$$

- *E.g.*,

```
p:Person ( int age = a,
           boolean isVoter = b
           || a >= 20, b == true, hasVoted(p))
```

Formalizing business rules—*Rule and ruleset*

- A ***rule*** is an expression of the form:

```
Rule ⟨name⟩:
  If ⟨patterns⟩
Then ⟨action⟩
```

- *E.g.*,

```
Rule VotingAge:
  If p:Person(age >= 18)
Then p.isVoter = true
```

- A ***ruleset*** is a finite set of rules.

Formalizing business rules—*Object pattern matching*

Let $\boldsymbol{o}$ be an object pattern:

$$\texttt{o:T(}T_1\ a_1\ \texttt{=}\ x_1\texttt{,}\quad \ldots,\quad T_n\ a_n\ \texttt{=}\ x_n$$

$$\| \, c_1\texttt{(}o\texttt{,}x_1\texttt{,}\ldots\texttt{,}x_n\texttt{)},\quad \ldots,\quad c_m\texttt{(}o\texttt{,}x_1\texttt{,}\ldots\texttt{,}x_n\texttt{))}$$

Let $\boldsymbol{o}'$ be an object instance:

$$o' = T'(\ T'_1\ a'_1 = v_1\ ,\ \ldots\ ,\ T'_n\ a'_n = v_{n'}\ )$$

Formalizing business rules—*Object pattern matching*

The pattern $\boldsymbol{o}$ *matches* the object instance $\boldsymbol{o}'$ iff:

- $\boldsymbol{T}' <: \boldsymbol{T}$
- $n \leq n'$
- $\forall i \in \{1, \ldots, n\}$, $\exists i' \in \{1, \ldots, n'\}$, $\boldsymbol{a}'_{i'} = \boldsymbol{a}_i$ and $\boldsymbol{T}'_{i'} <: \boldsymbol{T}_i$
- $\forall j \in \{1, \ldots, m\}$, $\boldsymbol{c}_j(\boldsymbol{o}, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ evaluates to **`true`**

Substitution $\sigma = \{\boldsymbol{o}'/\boldsymbol{o}, \boldsymbol{v}_1/\boldsymbol{x}_1, \ldots, \boldsymbol{v}_n/\boldsymbol{x}_n\}$ *realizes* the match

Formalizing business rules—*Rule action*

- **Working memory**: a (possibly empty) finite set of objects.
- **Rule Action**: a (possibly empty) sequence of one of:
  - $\langle$*location*$\rangle$ `=` $\langle$*expression*$\rangle$`;`
  - ***Assert*** $\langle$*object*$\rangle$`;`
  - ***Retract*** $\langle$*object*$\rangle$`;`

Formalizing business rules—*Rule application*

Given a rule action a, a substitution σ,

$$M' = \textbf{\textit{app}}^{\sigma}_{\textbf{a}}(M)$$

$$\textbf{\textit{app}}^{\sigma}_{\textbf{a}_1;\ldots;\textbf{a}_n}(M) \stackrel{\text{def}}{=} \begin{cases} M & \text{if } n = 0; \\ \textbf{\textit{app}}^{\sigma}_{\textbf{a}_2;\ldots;\textbf{a}_n}(\textbf{\textit{app}}^{\sigma}_{\textbf{a}_1}(M)) & \text{otherwise.} \end{cases}$$

$$\textbf{\textit{app}}^{\sigma}_{\texttt{x = e}}(M) \stackrel{\text{def}}{=} [e\sigma/x]M$$

$$\textbf{\textit{app}}^{\sigma}_{\texttt{Assert o}}(M) \stackrel{\text{def}}{=} M \cup \{o\sigma\}$$

$$\textbf{\textit{app}}^{\sigma}_{\texttt{Retract o}}(M) \stackrel{\text{def}}{=} M \setminus \{o\sigma\}$$

Formalizing business rules—*Application agenda*

Given a rule set S and a working memory M, define:

Agenda(S, M) : set of rule instances $\{\langle \rho_i, \sigma_i \rangle \mid i = 1, \ldots, n\}$ s.t. for all $i = 1, \ldots, n$,

- $\rho_i : p_i \longrightarrow a_i \in S$;
- $p_i = \langle o_{i1}, \ldots, o_{in_i} \rangle$;
- there exists $p'_i = \langle o'_{i1}, \ldots, o'_{in_i} \rangle \in M^{n_i}$ such that p_i matches p'_i with substitution σ_i.

Formalizing business rules—*BRMS interpretation scheme*

- let $S = \{R_i : P_i \rightarrow A_i \mid i = 1, \ldots, n\}$ be a ruleset,
- let $M_0 = \{o_j \mid j = 1, \ldots, m\}$ be an initial working memory:

[0] $M \leftarrow M_0$;
[1] $A \leftarrow$ ***Agenda***(S, M_0);
[2] While $A \neq \emptyset$ do:
[3]   ***Pick*** $\langle \rho = r : p \rightarrow a, \sigma \rangle \in A$;
[4] $M \leftarrow \textbf{\textit{app}}_a^\sigma(M)$;
[5] $A \leftarrow$ ***Agenda***(S, M);

Technical Issues—*Object models*

No accepted formal model of objects:

- BRMS object models vary wildy
- Pragmatics: *`CLOS`*, *C++*, *`Java`*, *C#*, *etc.*
- Web objects (RDF?)

Technical Issues—*Static analysis*

Rule validation is a critical issue for a BRMS

- It is hard enough to verify programs written by techies...
- It is even harder to verify rulesets specified by business folks!
- *Ad hoc* semantics (refraction, recency, priority, naming, *etc.*.)

Technical Issues—*Static analysis*

What properties may we wish to check for a ruleset?

- ***Liveness Properties***

In all executions, a (good) state is always reached:

- *The ruleset execution terminates.*
- *The premium is given a value.*
- *If Rule R_1 is fired, then Rule R_2 is always fired at some point afterwards.*

▶ *Safety Properties*

In all executions, a (bad) state is never reached:

- *This rule is never fired twice.*
- *The discount is never higher than 30%.*
- *The total budget will never be exceeded.*

Technical Issues—*Static analysis*

"Local" Properties

- Self-inconsistent rules; tautological conditions in rules.

- Coverage of enumerations, partitions, coverings.

- ***Non-Confluence***

Two (sequences of) rules, if executed in a different order, lead to incompatible states:

1. ***when*** *the shopping cart's value is more than $10* ***and*** *the customer is senior* ***then*** *set the discount to $10.*

2. ***when*** *the customer is Gold* ***then*** *set the discount to 10% of the shopping cart value.*

What about a senior Gold customer purchasing more than $10?

Technical Issues—*Static analysis*

For a shopping cart value v, the discounted value is $v' = v - \delta$, where δ is the discount.

$$(v \leq \$10 \;\wedge\; \delta_1 = \$0) \;\vee\; (v > \$10 \;\wedge\; \delta_1 = \$10)$$

by the first rule

$$\delta_2 = v \times 10\%$$

by the second rule

$$(v'_1 \leq \$10 \;\wedge\; \delta_{12} = \$0) \;\vee\; (v'_1 > \$10 \;\wedge\; \delta_{12} = \$10)$$

by the first rule

$$\delta_{21} = v'_2 \times 10\%$$

by the second rule

$$((v - \delta_1) \leq \$10 \;\wedge\; \delta_{12} = \$0) \;\vee\; ((v - \delta_1) > \$10 \;\wedge\; \delta_{12} = \$10)$$

by substitution

$$\delta_{21} = (v - (v \times 10\%)) \times 10\%$$

by substitution

Technical Issues—*Static analysis*

Once normalized, these constraints show that it is not inconsistent to have $\delta_{12} \neq \delta_{21}$.

For example, a senior Gold customer with a shopping cart of $\$15$ may pay either $\$3.5$ or $\$4.5$, depending on the order of rules.

Catching this non-confluence is provable only by non-trivial arithmetic constraint reasoning.

Perspectives—***Rule interchange***

BRMS vendors are keen on interchanging rules!

- ODMG PRR format
  recommendation pending

- W3C WG RIF (Horn rules + (E/)C/A rules)
  nowhere near completion! ...

- Rewrite rule community?
  where are you people? ... :-)

Perspectives—*Webization*

Need:

- Direct use of Web objects? (XML, RDF, “ontologies,” *etc.*.)
- Heterogeneous rulesets and object bases
- “Semantic Web?” ...

Conclusion

Business Rule Management Systems are a soaring market offering great research opportunities:

- Development of "Agile" applications (SOA/BPM)
- Non-conventional model of computation
- The rewrite-rule community ought to be more involved!

Thank You For Your Attention !

Email:

hak@ca.ibm.com
hak@acm.org