Improving BRMS Efficiency and Performance (And a bit on Conflict Resolution to help)

James C. Owen

KnowledgeBased Systems Corporation

Fort Worth, TX

jco@kbsc.com

Dr. Charles L. Forgy

Production Systems Technology

Pittsburgh, PA

clf@pst.com

Improving BRMS Efficiency and Performance (And a bit on Conflict Resolution to help)

A Review for those not totally familiar with BRMS, DSS and RBS

BRMS: (Business Rules Management System) Sometimes referred to as BDMS – Business Decision Management System. A formalized collection of decisions (which leads to a set of rules) and facts that closely follows the decisions that have to be made by any business enterprise, whether in ARx (Accounts Receivable), AP (Accounts Payable), GL (General Ledger) or any other facet of the business enterprise.

This can be shown to the user as a Decision Table (think Excel), Decision Tree, Decision Map or IF-THEN-ELSE rules. These formats are sometimes called metaphors, meaning the external GUI that manages the decisions and rules. Yet, all of them should be synchronized so that a change in one representation is reflected in the others.

Improving BRMS Efficiency and Performance (And a bit on Conflict Resolution to help)

A Review for those not totally familiar with BRMS, DSS and RBS

RBS: **(RuleBased System) A collection of rules (productions) and facts that simulate the human brain in a very narrow area of expertise that mimics what a human domain expert might do.**

- **Increased availability (24x7x52)**
- **Reduced Costs.**
- **Permanent.**
- **Reliability (same answer all the time).**
- **Faster response than a human.**
- **Expertise can be incorporated from many experts.**
- **Never sick. Never takes a day off. Never retires.**
- **Not emotional: Doesn't get happy. Doesn't get sad**

Improving BRMS Efficiency and Performance (And a bit on Conflict Resolution to help)

A Review for those not totally familiar with BRMS, DSS and RBS

Initial Questions to Ask

- Is the domain "well bounded"?
- Is there a real need for an ES?
- Is there at least ONE human expert?
- Can the DE explain the methods to the KE?
- Can the problem be solved "more easily" by conventional programming. (An ES with a lot of control code usually disguises a very specific algorithm.)
- Is the problem heuristic and uncertain? (??)

Improving BRMS Efficiency and Performance (And a bit on Conflict Resolution to help)

A Review for those not totally familiar with BRMS, DSS and RBS

Selecting the proper BRMS tools or method: (per BFKM, p 464) **BC** is Backward Chaining and **FC** is Forward Chaining.

- **Diagnostic problems : BC**
- **Monitoring Problems : FC**
- **Interpretation : FC**
- **If the nature of the data are well understood but goals are not clearly defined , use FC.**
- **If there is a goal (goals or sub-goals) then BC is MUCH better.**
- **If everything is clearly defined, then use something OTHER than a BRMS (COBOL, Java, C/C++, something procedural)**

Improving BRMS Efficiency and Performance (And a bit on Conflict Resolution to help)

A Review for those not totally familiar with BRMS, DSS and RBS

Two Concepts (Remember)

First, a BRMS (rulebase) is a DECLARATIVE language, not procedural like C, C++ nor Java. (OO is still procedural, not declarative.)

Second, a rulebase is NON-MONOTONIC. This means that the Agenda (rules which are true and can fire) can grow or shrink with the rules and the data. Dr. Forgy gave it the following definition:

Improving BRMS Efficiency and Performance (And a bit on Conflict Resolution to help)

A Review for those not totally familiar with BRMS, DSS and RBS

[From an email dated 4 April 2005]

I can sort of guess about the rest of this example, and I would have to say that it is non-monotonic reasoning--if maybe not the best example. The essence of non-monotonicity, as I understand it, is that adding new information can cause previous conclusions to become invalid. (In a monotonic logic, adding information can only cause the set of conclusions to grow.)The canonical example, is probably Prolog's negation-as-failure: Eg, the rule

```
a :- b, c, not(d)
```

would conclude a is true if b and c are known, but d is not known. If d was added later, the conclusion a would then be wrong.

-- CLF

Improving BRMS Efficiency and Performance (And a bit on Conflict Resolution to help)

1. **We are NOT going to discuss Rete**
2. Avoid LHS (CEs) that match many WMEs
3. Avoid big cross-products between conditions
4. Avoid frequent changes to matched conditions
5. Make matching individual CEs faster
6. Limit the size of the Conflict Set
7. Call user-defined functions within the RBS

Improving BRMS Efficiency and Performance

- Order of development should be something like this:
  1. Initialization Rules (setup)
  2. Knockout Rules (must be 18 for car insurance)
  3. Rule Flow (normal business flow with nodes for sets of rules)
  4. Clean-up rules (deletion of persistent objects…)
  5. Output results

Improving BRMS Efficiency and Performance

Limit the number of elements that match a condition

- Limit the range of elements by physical property
- Migrate from very general rules to more smaller rules that will yield less processing time
- Limit the range of elements by processing state

Improving BRMS Efficiency and Performance

The MEA (Means-Ends Analysis) Conflict Resolution strategy allows complete control over context processing

- Create separate element classes
- Represent and enumerate sequence carefully
- [long discussion here]

Originally: When Dr. Charles Forgy and his co-worker, Dr. John McDermott, were working on the original OPS2 system for DEC, Dr. McDermott was explicitly ordering goals in the early version of R1. So Dr. Forgy modified LEX to create MEA. MEA then became the preferred method for conflict resolution in all subsequent OPS systems. (Per email from Dr. Forgy on 1 July 2003.)

Improving BRMS Efficiency and Performance

- Avoid Big Cross Products (KISS)
  - Reorder CEs (*More on this later)*
  - Avoid big conditional elements
  - Add new attributes in the rules ONLY (*not external objects*)
  - Repeat attributes when possible
  - Merge element classes
  - [OPSJ, CLIPS and Jess] Use the "build" command to specialize the rules to create more rule sets

Improving BRMS Efficiency and Performance

Avoid frequent changes to matched conditions

- Place rapidly-changing elements last in the LHS
- Avoid excessive changes in the CEs
  - Avoid changes in unrelated attributes
  - Carefully consider deleting old elements
  - Reduce the number of control elements by composition

Improving BRMS Efficiency and Performance

Match Individual CEs more quickly

- Place restrictions tests early
- Change the representation of the data
- Most-Specific Patterns go first *(G&R-9.13)*
- Patterns matching volatile facts go last *(G&R-9.13)*
- Patterns matching the fewest facts go first *(G&R-9.13)*

Improving BRMS Efficiency and Performance

Limit the size of the Conflict Set

- Large Conflict Sets = inefficiency
- Ensure some kind of time limit for processing a rule
- DO NOT use recency nor specificity for control flow. *(Yes, I know that this is the "preferred" method for some BRMS programmers.)*
- Use Goal-Oriented Programming to control flow when necessary (Explain if needed)
- Use the "correct" conflict resolution method ☺ Normally, today, this is not available except with CLIPS and a few others.

Improving BRMS Efficiency and Performance

- Call user-defined functions from within rules
- Enable/Disable rules
  - Date ranges
  - Time ranges
- Use Goal-Oriented programming to
  - Control Rules
  - Sub-Goaling Rules
  - Rules for immediately soluble goals

Improving BRMS Efficiency and Performance

Control Strategies (Advantages)

- Processing is usually faster
- Data elements or goals are treated uniformly by the RBS
- Programming time is not wasted writing repetitive control constructs
- Many editing errors can be prevented

Improving BRMS Efficiency and Performance

Control Strategies (MYCIN S&B, 558)

- Put non-self-referencing rules first (*Ex later)*
- If nothing has been observed, consider situations that have no visible manifestation.
- If unable to make a decision, assume the most probable situation
- If there is evidence for two hypothesis, A and B, that tend to be confused, then rule out B.
- for more on this subject see p.578 at http://people.dbmi.columbia.edu/~ehs7001/Buchanan-Shortliffe-1984/MYCIN%20Book.htm

Improving BRMS Efficiency and Performance

Example of self-referencing rule to save for last

If

tetracycline will NOT kill the patient because the patient is between 8 and 18 years old and in good health

THEN

Rx tetracycline

Conflict Resolution

A **conflict-resolution strategy** is a coordinated set of principles for selecting among competing instantions on the Agenda Table that contains all of the rules that are true during this cycle. BRKM, p304

An expert system's performance depends on the conflict resolution for both *sensitivity and stability*: (McDermott & Forgy '78)

Sensitivity is the systems ability to respond to dynamic changes to the surrounding environment. MEA gives sensitivity to the system because it uses time-stamp codes to control rule order firing.

Stability is the "continuity of behavior." Often this is at odds with the Sensitivity part of the system.

Improving BRMS Efficiency and Performance

Conflict Resolution

Original Rete Conflict Resolution cica 1979

1. **Halt the system if conflict set *(Agenda)* is empty.**
2. **Exclude from consideration all instantiations that have previously executed.**
3. **Order instantiations based on recency of data elements the contain then exclude all instantiations except the ones dominating under this order. Follow the most recent time stamp for any tie situations.**
4. **Select the rule with the most Condition Elements.**
5. **Select the rule with the most constant atoms in the LHS.**
6. **Select the most recently created production.**
7. **Make an arbitrary selection.**

See J. McDermott and C. Forgy, "Production System Conflict Resolution Strategies" in Pattern-Directed Inference Strategies, D. A. Waterman and F. Hayes-Roth, Ed., Academic Press, New York, 1978, p. 177-199

Conflict Resolution

Rule Conflict Resolution in **Some** BRMS

Note: Some systems use a variant of the MEA (Means Ends Analysis) strategy from OPS but still add Rule Class and Rule Priority

1. Refraction: Eliminate all rules which have previously fired where a rule is defined as the logic PLUS the specific data.

2. Rule Class: Select rule with highest class number.

3. Context Recency: Select the rules that match the most recent objects using the objects time stamp.

4. Rule Priority : Select rule with highest priority number.

5. Rule Specificity: Select the rule with the greatest number of condition elements in the LHS.

6. Arbitrary. (Probably the one that comes first in the text... :-)

Conflict Resolution

Difference between MEA and LEX Strategies

In the step on Recency, LEX considers only the highest time stamp of rule in general. However, in MEA this is a two-part resolution:

MEA Time Stamp Considerations:
1. Order the instantiations based on the recency of the ***first*** time tag. Select the most recent instantiation.
2. If more than one instantiation shares the highest first time tag, order these instantiations based on the recency of ***all the time tags*** in the instantiation. Select the most recent instantiations.

Note: In his original thesis on Rete (p.8) Dr. Forgy used the LEX method of conflict resolution rather than MEA. Later he highly recommended using MEA using the McDermott paper as his reasoning.

(Forgy-79) Charles L. Forgy; "On the Efficient Implementation of Production Systems"; Ph.D. Thesis; Carnegie-Mellon University; Feb, 1979; 178 pp.
(Forgy-82) Charles L. Forgy; " Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem "; *Artificial Intelligence*, 1982, pp. 17-37.
(McDermott-78) John McDermott and Charles L. Forgy; "Production System Conflict Resolution Strategies"; Pattern-Directed Inference Systems, D. A. Waterman and F. Hayes-Roth, Eds., Academic Press, pp. 177-199.

Conflict Resolution

Rules on the Agenda Table and associated time stamps, assuming that all rules have the same specificity:

		MEA	LEX
1.	rule A:	10, 9, 6, 43	10, 9, 6, 43
2.	rule A:	10, 43, 9, 3	10, 43, 9, 3
3.	rule B:	12, 23, 17	12, 23, 17
4.	rule C:	12, 9, 23	12, 9, 23
5.	rule D:	9	9
6.	rule D:	12	12

Color coding: yellow, purple, green, red. Tie is underlined.

MEA: Positions 3, 4 & 6 share 1st ts. of 12. Pos 3, rule A and pos 4, rule C share a 2nd highest ts. of 23. Rule B in pos 3 will fire because it has a 3rd highest ts. of 17. (then positions 4, 6, 1, 2, 5)

LEX: Pos 1, rule A & pos 2, rule A share time stamps (ts.) of 43, 10 and 9. Pos 1, rule A will fire because ts. 6 is higher than ts. 3. (then 2, 3, 4, 6, 5)

Conflict Resolution

Questions ?
Comments ?
Suggestions ?

Improving BRMS Efficiency and Performance

EXTRA INFORMATION FOR GEEKS

Improving BRMS Efficiency and Performance

Girratano and Riley (Para 9.12) pointed the importance of pattern order (the order of the Ces) in a rule. Consider that if we have the following facts:

```
(defacts information
    (find-match a c e g)
    (item a)
    (item b)
    (item c)
    (item d)
    (item e)
    (item f)
    (item g)  )
```

(For those who do not read OPS, this will be re-written in Java later)

Improving BRMS Efficiency and Performance

Consider that if we have the following rule:

```
(defrule match-1
    (item ?x)
    (item ?y)
    (item ?z)
    (item ?w)
    (find-match ?x ?y ?z ?w)
=>
(assert  (found-match ?x ?y ?z ?w )
```

This is the same rule EXCEPT that the (find-match…) has been moved to the last CE rather than the first.

Improving BRMS Efficiency and Performance

Consider that if we have ANOTHER rule:

```
(defrule match-2
    (find-match ?x ?y ?z ?w)
    (item ?x)
    (item ?y)
    (item ?z)
    (item ?w)
=>
(assert  (found-match ?x ?y ?z ?w )
```

(For those who do not read OPS, this will be re-written in Java later)

Improving BRMS Efficiency and Performance

Using the "watch facts" command in CLIPS (or Jess) Rule 1 has 5 pattern matches and **29** partial matches and runs fairly quickly.

However, Rule 2 will have 5 pattern matches and, because of the way that it is written, it will have **2,801** partial matches. On older machines, it will run quite slowly. On my brand-new MacLapBookPro (16 GB of RAM, quad i7) or my Toshiba that has an identical setup, it will run too fast to watch. So, you have to add a lot more objects, which is fairly easy using cut-and-paste.

(For those who do not read OPS, this has been re-written in Java. See the accompanying white paper for the Java version.)

Reference Material

- **Joseph Girratano and Gary Riley**; **Expert Systems Principles and Programming**; [1st Ed. 1989, 2nd Ed. 1994, , 3rd Edition 1998, 4th Ed. 2005] PWS Publishing Company
- **Bruce G. Buchanan and Edward H. Shortliffe**; **Rule-Based Expert Systems – The MYCIN Experiments of the Stanford Heuristic Programming Project**, 2nd Edition; Addison-Wesley Publishing
- **John McDermott and Charles L. Forgy**; **Production System Conflict Resolution Strategies**; Pattern-Directed Inference Systems, D. A. Waterman and F. Hayes-Roth, Eds., Academic Press, pp. 177-199
- **Lee Brownston, Robert Farrell, Elaine Kant, Nancy Martin**; **Programming Expert Systems in OPS5**, Addison-Wesley Publishing
- **John Durkin**; **Expert Systems Design and Development**, Macmillan Publishing,

Reference Material

- **Stacy Joines, Ruth Willeborg and Ken Hygh**; **Performance Analysis for Java Websites**; Addison-Wesley
- **Thomas A. Cooper and Nancy Wogrin**; **Rule-based Programming with OPS5**, Morgan Kaufman Publishers
- **Jérome Boyer and Hafedh Mil**i; **Agile Business Rule Development** (ABRD) Process, Architecture and JRules Examples, Springer Press (Highly recommended for ODM projects)
- **Hafedh Mili, Ali Mili, Sherif Yacoub, Edward Addy**; **Reuse-Based Software Engineering** – Techniques, Organization and Controls; Wesley-Interscience