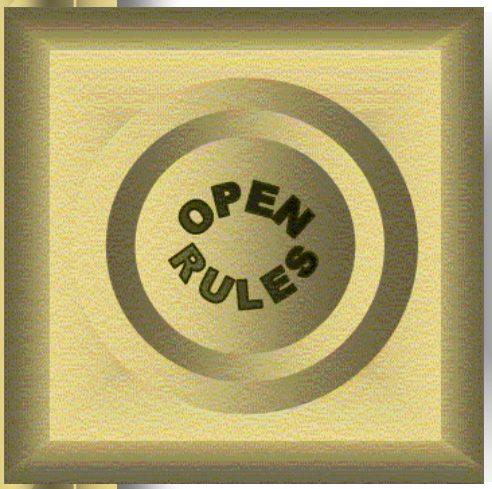

Practical Techniques for Resolving Conflicts among Business Rules

Presenter: Dr. Jacob Feldman

OpenRules Inc., CTO

jacobfeldman@openrules.com

www.OpenRules.com

Motivation

- Contradictory business rules occur in normal business situations, and maintaining rules with exceptions is a very typical example of rule conflicts
- In real-world of complex decision modeling, business analysts frequently face issues related to diagnostic and resolution of business rule conflicts
- To avoid conflicts, business analysts have to add more and more rules making their maintenance a real problem

Real-world Examples

- **No vehicles in the park** (except during parades)
- **Offer, acceptance, and memorandum produce a contract** (except when the contract is illegal, the parties are minors, inebriated, or incapacitated, etc.)
    - These rules are “defeasible” as they can be defeated by their exceptions

Example from Financial Domain:

- **Rule 1:** Stock in debt is considered risky
- **Rule 2:** Stock in fusion with other stocks may be risky
- **Rule 3:** Stock in fusion with a strong stock is not risky
- **Rule 4:** Do not buy risky stocks unless they have a good price

Questions

- *What are the commonly used techniques for resolving rule conflicts?*
- *Is it possible to automatically resolve rule conflicts?*
- We will discuss how traditional and modern BRMS systems address these questions

Example of Rules with Conflicts

- **Rule 1: Birds can fly**
- **Rule 2: Chicken cannot fly**
- **Rule 3: Scared chicken can fly**

- Even little children can apply these rules in many practical situations
- How will a BRMS represent these rules?

Example of Rules with Conflicts within a BRMS

Decision table that avoids rule conflicts by considering ALL "if-then" combinations in a mutually exclusive way:

DecisionTable DefineAbilityToFly							
Condition		Condition		Condition		Conclusion	
Bird		Chicken		Scared		Ability To Fly	
Is	Yes	Is	No			Is	Yes
Is	Yes	Is	Yes	Is	No	Is	No
Is	Yes	Is	Yes	Is	Yes	Is	Yes
Is	No					Is	?

- Rule 1: Birds can fly
- Rule 2: Chicken cannot fly
- Rule 3: Scared chicken can fly

All possible If-Then-Else Combinations Become a Challenge

- **Rule 1: Birds can fly**
- **Rule 2: Chicken cannot fly**
- **Rule 3: Scared chicken can fly**

Add two more rules:

- **Rule 4: Penguins cannot fly**
- **Rule 5: Everybody can fly in the airplane**

Expanded Single-Hit Decision Table

DecisionTable DefineAbilityToFly												
Condition		Condition		Condition		Condition		Condition		Conclusion		Message
Bird		Chicken		Scared		Penguin		In Airplane		Ability To Fly		Message
								Is	Yes	Is	Yes	
Is	Yes	Is	No			Is	No	Is	No	Is	Yes	
Is	Yes					Is	Yes	Is	No	Is	No	
Is	Yes	Is	Yes			Is	Yes	Is	No	Is	?	Chicken cannot be penguin
Is	Yes	Is	Yes	Is	Yes	Is	No	Is	No	Is	Yes	
Is	Yes	Is	Yes	Is	No	Is	No	Is	No	Is	No	
Is	No							Is	No	Is	?	We don't know if non-bird can fly

If you try to cover all possible combinations for similar rules with conflicts, the number of rules grows exponentially!

- Rule 1: Birds can fly
- Rule 2: Chicken cannot fly
- Rule 3: Scared chicken can fly
- Rule 4: Penguins cannot fly
- Rule 5: Everybody can fly in the airplane

Problems with Traditional Single-Hit Decision Tables

- Difficult to read and understand such a decision table not mentioning a necessity to maintain it with future changes
- Think about adding new rules:
  - **Birds with broken wings cannot fly**
  - **Ostriches would not fly even when they are scared**
- What if we try to add more rules that cover other 40 kinds of flightless birds that are in existence today?

Switching to Multi-Hit Decision Tables

DecisionTableMultiHit DefineAbilityToFly

Condition		Condition		Condition		Condition		Condition		Conclusion		Message
Bird		Chicken		Scared		Penguin		In Airplane		Ability To Fly		Message
Is	Yes									Is	Yes	
Is	Yes	Is	Yes							Is	No	
Is	Yes	Is	Yes	Is	Yes					Is	Yes	
Is	Yes					Is	Yes			Is	No	
								Is	Yes	Is	Yes	
Is	No							Is	No	Is	?	We don't know if non-bird can fly

Multi-Hit Decision Tables allow **Rules Overrides**:

Rules with more specific conditions may override previously defined rules with more generic conditions!

- Rule 1: Birds can fly
- Rule 2: Chicken cannot fly
- Rule 3: Scared chicken can fly
- Rule 4: Penguins cannot fly
- Rule 5: Everybody can fly in the airplane

Pros and Cons of Multi-Hit Decision Tables

- Pros:
  - more readable and easy to maintain to compare with single-hit decision tables
  - you do not have to cover all possible combinations of decision variables
- Cons:
  - relies on a strict sequencing of the rules inside the decision table that makes an introduction of new concepts and rules much more problematic

Auto-Resolution of Rules Conflicts

- *Is it even possible to automatically resolve conflicts between business rules?*
- The closest theory that deals with business rule conflicts is known as "Defeasible Logic" introduced more than 25 years ago
- This is a kind of reasoning that is based on reasons that are *defeasible*, i.e. capable of being defeated by other reasons

Defeasible Logic

- Differentiates between strict rules and defeasible rules:
  - ***Strict rules*** are rules in the classical sense that are used in all modern BRMSs, e.g. "*If something is a penguin Then it is a bird*".
  - ***Defeasible rules*** are rules that can be defeated by contrary evidence, e.g. "*Birds typically can fly unless there is other evidence suggesting that it may not fly*".
  - ***Defeaters*** are special rules used only to defeat some defeasible rules, e.g. "H*eavy animals may not be able to fly*".

Superiority Relations among Rules

- Used to define priorities among rules, where one rule may override the conclusion of another rule.
- For example, given the defeasible rules
  - ***R1****: Birds typically fly*
  - ***R2****: Birds with broken wings cannot fly*

  no conclusive decision can be made about whether a bird with broken wings can fly.

  But if we introduce a superiority relation

  R2 > R1

  then we can indeed conclude that it can't fly.

Defeasible Logic & BRMSs

- The majority of BRMSs do not support the defeasible logic forcing their users to resolve all conflicts manually
- However, today enterprise-level rule repositories achieved a high level of maturity and internal complexity
- Absence of automatic conflict resolution tools will lead to unnecessary growth of rules and may gradually convert rules repositories to unmaintainable “monsters”
- Defeasible Logic becomes a must-feature

Open Source BR Products take a Lead

- Last year two major open source BRMSs announced their implementations of the Defeasible Logic:
  - JBoss Drools implemented the classic defeasible logic with strict and defeasible rules along with the superiority relationships between rules
  - OpenRules implemented the defeasible logic with strict and defeasible rules but using a different concepts for conflict resolution based on their constraint-based rule engine

Defeasible Logic by Drools

- JBoss Drools added the following rule annotations to their rule language (DRL):
  - *@Strict*
  - *@Defeasible*
  - *@Defeats(“rule1", “rule2", “rule3”)*
  - *@Defeater*
- A user may use *@Defeats* to specify a list of defeasible rules that can be defeated by the current rule

Defeasible Logic by Drools

- Rules for issuing bus tickets in DRL:

```
rule "Do not issue to banned people" @Defeasible when
  p : Person( )      Banned( person == p )
then
  logicalInsert(new ChildBusPass( p ) , “neg” );
end

rule "Exception for children with minor offences" @Defeats("Do not issue to banned people") when
  p : Person( )
      IsChild( person == p )
      Banned( person == p, offence == "minor" )
then
  logicalInsert(new ChildBusPass( p ) );
end
```

- Read more

Defeasible Logic by OpenRules

- While we could also list rules that can be defeated by the current rules, we believe such “superiority relations” will become unmaintainable after a while
- If some rules directly “know” about other rules it may lead to “macaroni” relations especially when new defeasible rules need to be added

Defeasible Logic by OpenRules

- To implement Defeasible Logic without "superiority relations", we introduced a "**rule probability**" (or rule likelihood)
- It means instead of stating "*Birds typically can fly*" our user is able to write something like:
  - *There is a "relatively high" probability that birds can fly* *(defeasible rule)*
  - *There is a "very high" probability that penguins cannot fly* *(defeasible rule)*
  - *Everybody can fly in an airplane* *(strict rule)*

Defeasible Logic by OpenRules

- Our user may assume that Rule with a higher probability will in general defeat Rule with a smaller probability
- The rule probabilities may be expressed as:

  NEVER, VERY LOW, LOW, BELOW MID, MID, ABOVE MID, HIGH, VERY HIGH, ALWAYS
- Or using numbers 0 (NEVER), 1, 2, …, 99, and 100% (ALWAYS)
- A rule with probability ALWAYS (or not specified) means a strict rule

Defeasible Logic by OpenRules

- Actually we added *only one* optional column to our standard decision table template called "***ActionProbability***"

DecisionTable DefineAbilityToFly												
Condition		Condition		Condition		Condition		Condition		ActionProbability	Conclusion	
Bird		**Penguin**		**Chicken**		**Scared**		**In Airplan**		**Probability**	**Ability To Fly**	
Is	Yes									MID	Is	Yes
		Is	Yes							VERY HIGH	Is	No
								Is	Yes		Is	Yes
				Is	Yes					HIGH	Is	No
				Is	Yes	Is	Yes			VERY HIGH	Is	Yes
Is	No							Is	No		Is	?

- Rule designer should be careful defining relative probabilities. For example, if we forget to specify the condition "*Chicken is Yes*" in the rule 5, it would not be clear either a scared penguin can fly or not.

How Does It Work

- The described logic is supported by OpenRules Engine known as "Rule Solver" that is based on a standard constraint solver
- Rule Solver creates a constraint satisfaction problem:
  - Constraints for all "strict" rules are simply posted as hard constraints
  - Constraints for all rules with probabilities are posted as soft constraints with a possible violation cost defined by its probability value
- Then Rule Solver automatically solves this problem by **minimizing the total constraint violation** for all defeasible rules

Benefits

- The described approach will work even when not all conflicts can be resolved: the Rule Solver will find a decision with minimal total conflicts
- Business analysts may express their preferences in an intuitive way as they do it in everyday life when they say:

  "*There is a high probability of rain tonight*"

  without any knowledge of the defeasible logic or the probability theory

Conclusion

- We did not want to create a false impression that all problems related to rule conflicts have been solved
- Our objective was to bring an attention to the importance of these issues and to show some possible ways for their resolution
- We expect that all major BR vendors gradually will add an automatic ability to solve rule conflicts to their product offerings