RuleML 2011
The 5th International Symposium on Rules: Research Based and Industry Focused

# Representing and Solving Rule-based Decision Models with Constraint Solvers

Jacob Feldman, Ph.D.

Founder & CTO, OpenRules, Inc.
JSR-331 Specification Lead

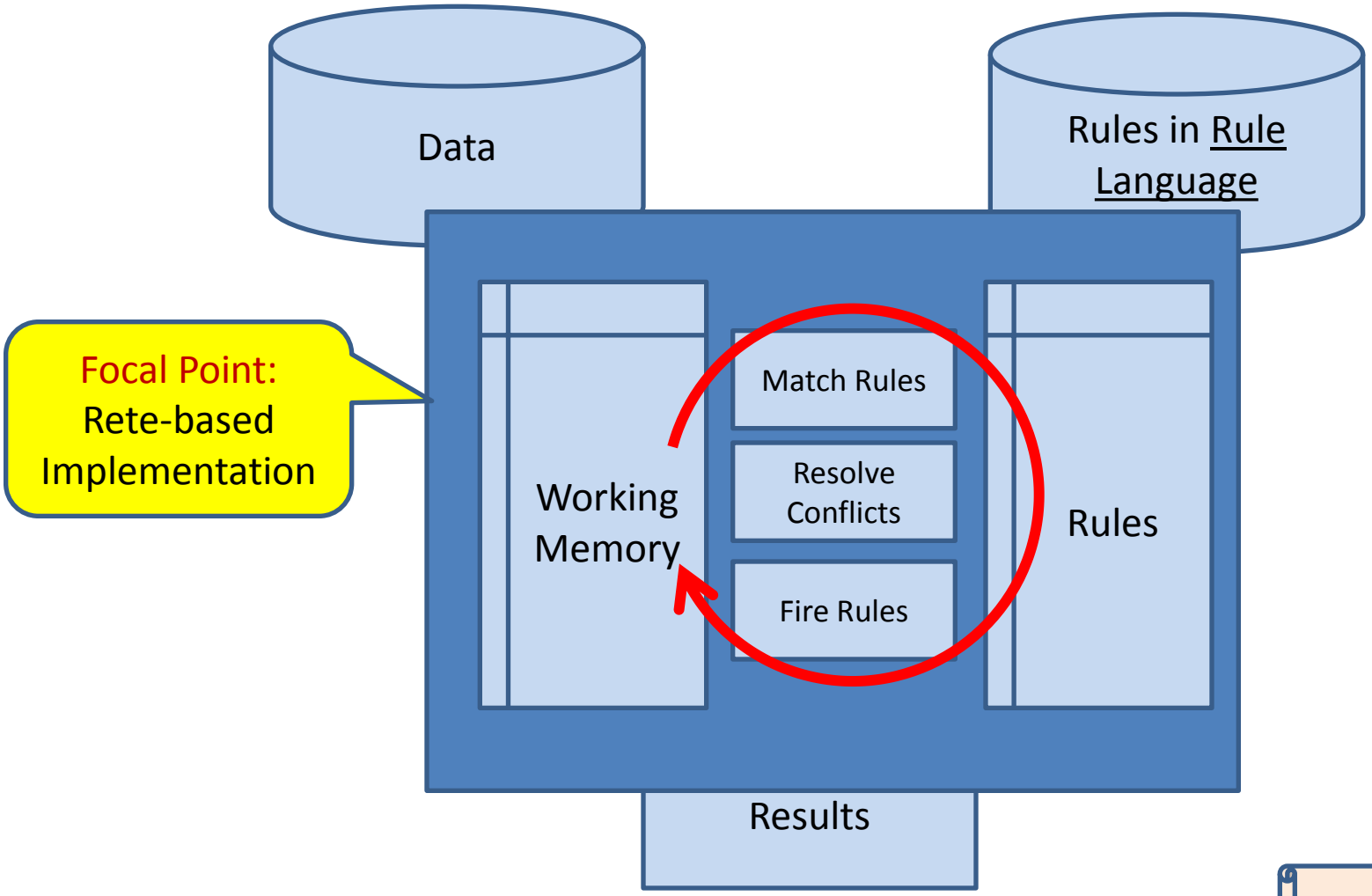
RuleML
Realize your Knowledge

OPEN RULES

# Two Types of Rule Engines

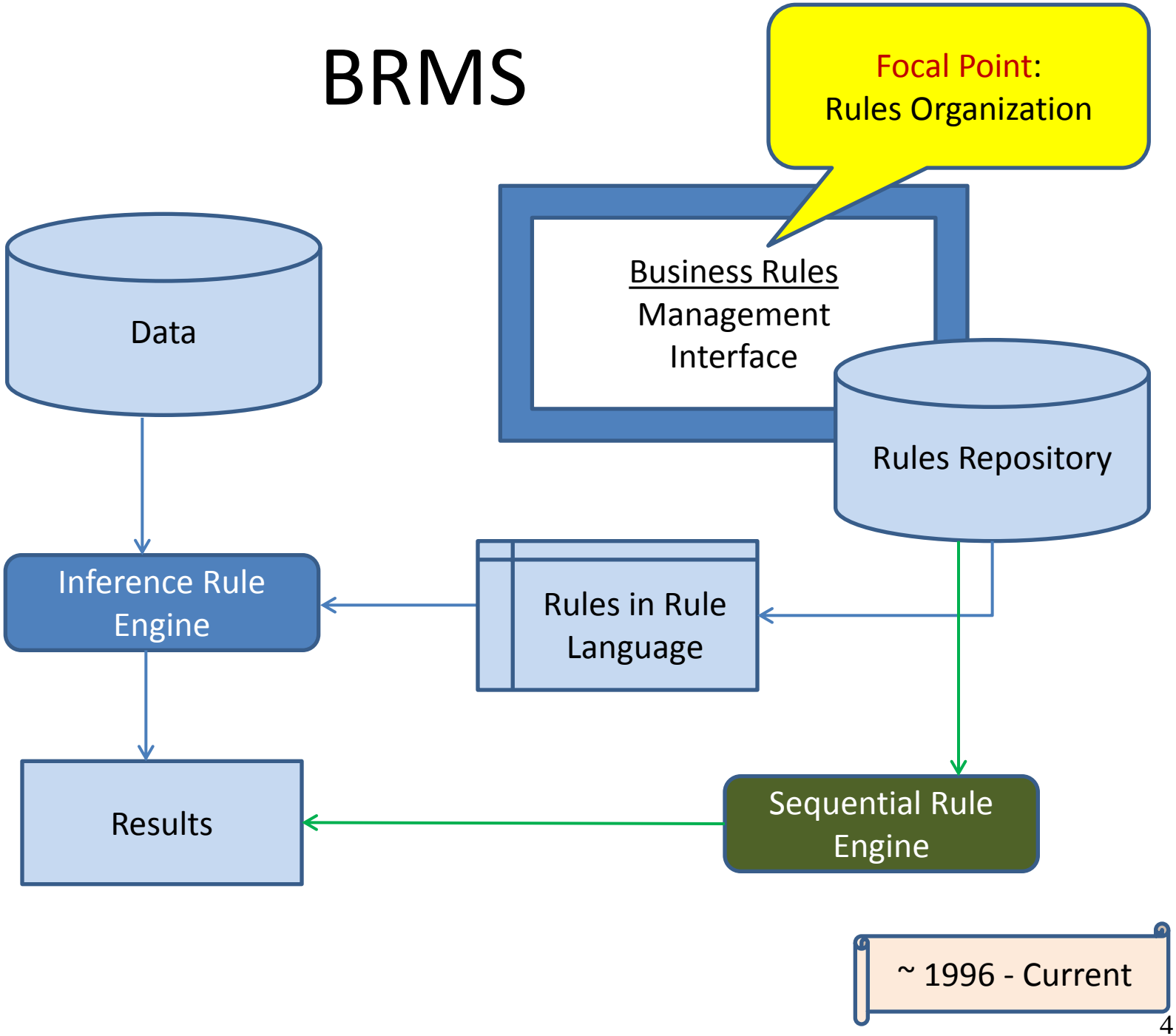
- **Inferential** Rule Engines
  - support a pure declarative representation of business rules
  - Rete-based
- **Sequential** Rule Engines
  - rely on user-defined sequencing of rules and rule families

# Rule Engines



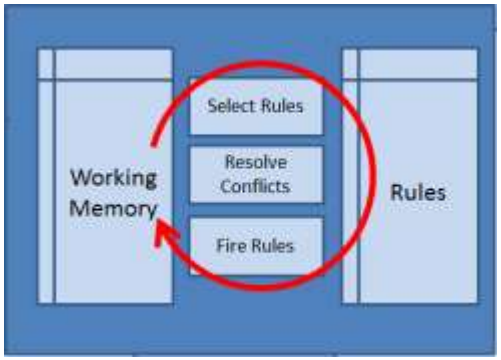

~ 1982-1995

# BRMS



# 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
- Decision Modeling  without a rule language:
  - OMG Standard "DMN" (Decision Modeling and Notation)

~ 2007 - Current

# Rule Family Examples

RuleFamily PersonLikelihoodOfDefaultingOnLoan

| Condition | | Condition | Condition | Condition | Conclusion | |
|---|---|---|---|---|---|---|
| Person Employment History | | Person Mortgage Situation | Person Miscellaneous Loans Assessment | Person Outside Credit Score | Person Likelihood of Defaulting on a Loan | |
| Is | Po | | | | Is | High |
| Is | Go | | | | Is | Low |
| Is | Po | | | | Is | Medium |
| | | | | | Is | High |

RuleFamily PersonEmploymentHistory

| Condition | | Condition | | Conclusion | |
|---|---|---|---|---|---|
| Person Years at Current Employer | | Person Number of Jobs in Past Five Years | | Person Employment History | |
| < | 1 | > | 5 | Is | Poor |
| < | 1 | <= | 5 | Is | Average |
| Within | [1;2] | > | 5 | Is | Poor |
| Within | [1;2] | <= | 5 | Is | Average |
| > | 2 | <= | 4 | Is | Good |
| > | 2 | Within | (4;6] | Is | Average |
| > | 2 | > | 6 | Is | Poor |

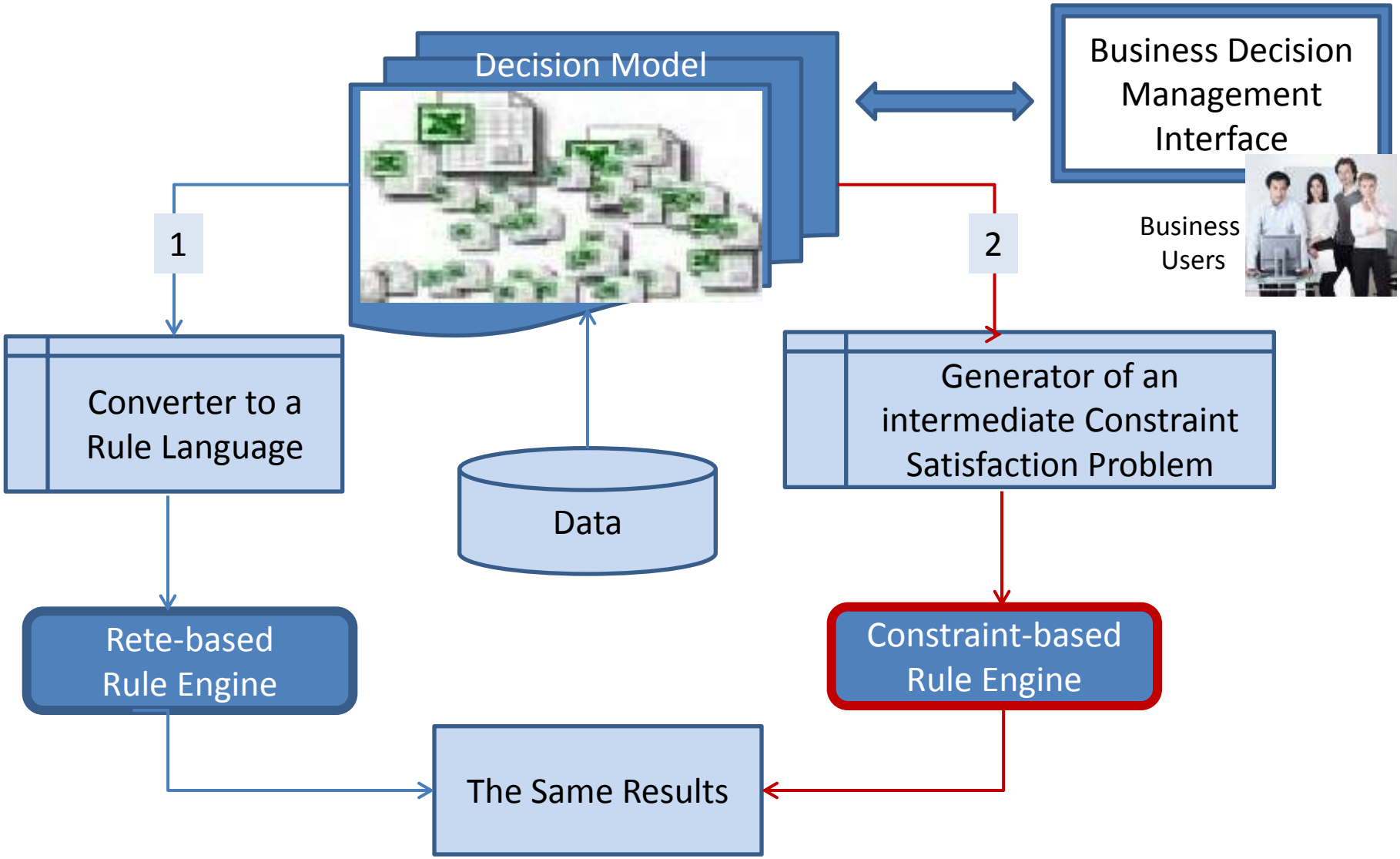
# Rules: To Order or Not to Order

- Decision Modeling 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



# JSR-331 “Constraint Programming API”

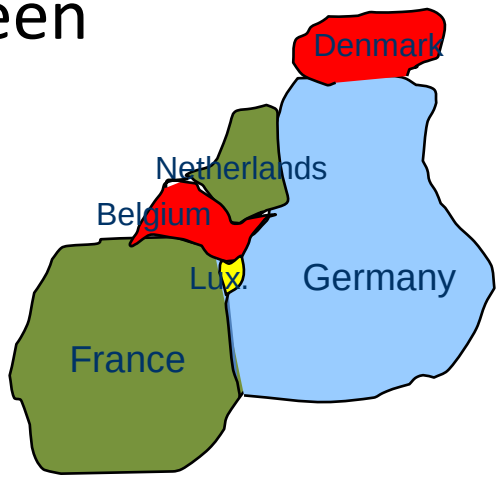
- JSR 331 is “Constraint Programming API” standard developed within the Java Community Process (JCP) www.jcp.org
- JSR-331 covers key concepts and design decisions related to the standard representation and resolution of constraint satisfaction and optimization problems
- Reached The Final Draft phase
- Currently has 3 working implementations

# JSR-331 Key Objectives

- Make CP more accessible for business software developers
- Allow a Java business application developer to easily switch between different solver implementations <u>without any changes</u> in the application code
- Make CP a foundation for creating different inference engines

# CSP Sample: “Map Coloring”

- A map-coloring problem involves choosing colors for the countries on a map in such a way that at most 4 colors are used and no two neighboring countries have the same color
- We will consider six countries: Belgium, Denmark, France, Germany, Netherlands, and Luxembourg
- The colors are blue, white, red or green

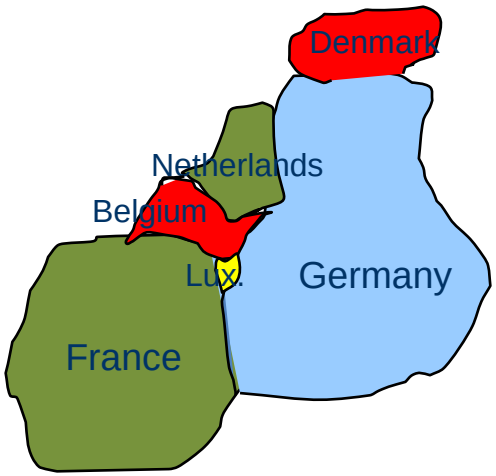


# Example “Map Coloring”: problem variables

```
static final int MAX = 4;  // number of colors
Problem p = ProblemFactory.createProblem(“Map”);
// Constrained Variables
Var Belgium      =   p.variable("Belgium“,      0, MAX - 1);
Var Denmark      =   p.variable("Denmark“,      0, MAX - 1);
Var France       =   p.variable ("France“,      0, MAX - 1);
Var Germany      =   p.variable ("Germany“,     0, MAX - 1);
Var Netherlands =  p.variable ("Netherlands“,0, MAX - 1);
Var Luxemburg  =   p.variable ("Luxemburg“,  0, MAX - 1);
```

Each country is represented as a variable that corresponds to an unknown color: 0,1,2, or 3

# “Map Coloring”: problem constraints



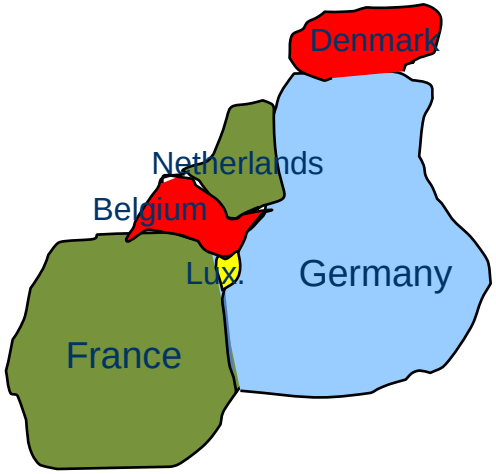

```
// Define Constraints
France.ne(Belgium).post();
France.ne(Luxemburg).post();
France.ne(Germany).post();
Luxemburg.ne(Germany).post();
Luxemburg.ne(Belgium).post();
Belgium.ne(Netherlands).post();
Germany.ne(Netherlands).post();
Germany.ne(Denmark);
```

```
// We actually create a “not-equal” constraint and then post it
Constraint c = Germany.ne(Denmark);
c.post();
```

# “Map Coloring”: solution search

```
// Solve
Solution solution = p.findSolution();
if (solution != null)  {
   solution.log()
}
```

```
// Solution:
Belgium – red
Denmark – red
France – green
Germany – blue
Netherlands – green
Luxemburg - yellow
```



# 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:
  - Decision Model as a common input for CP and BR engines
  - Decision Modeling Notation (DMN) – standardization efforts
  - JSR-331 provides a standard API for different Java-based CP solvers

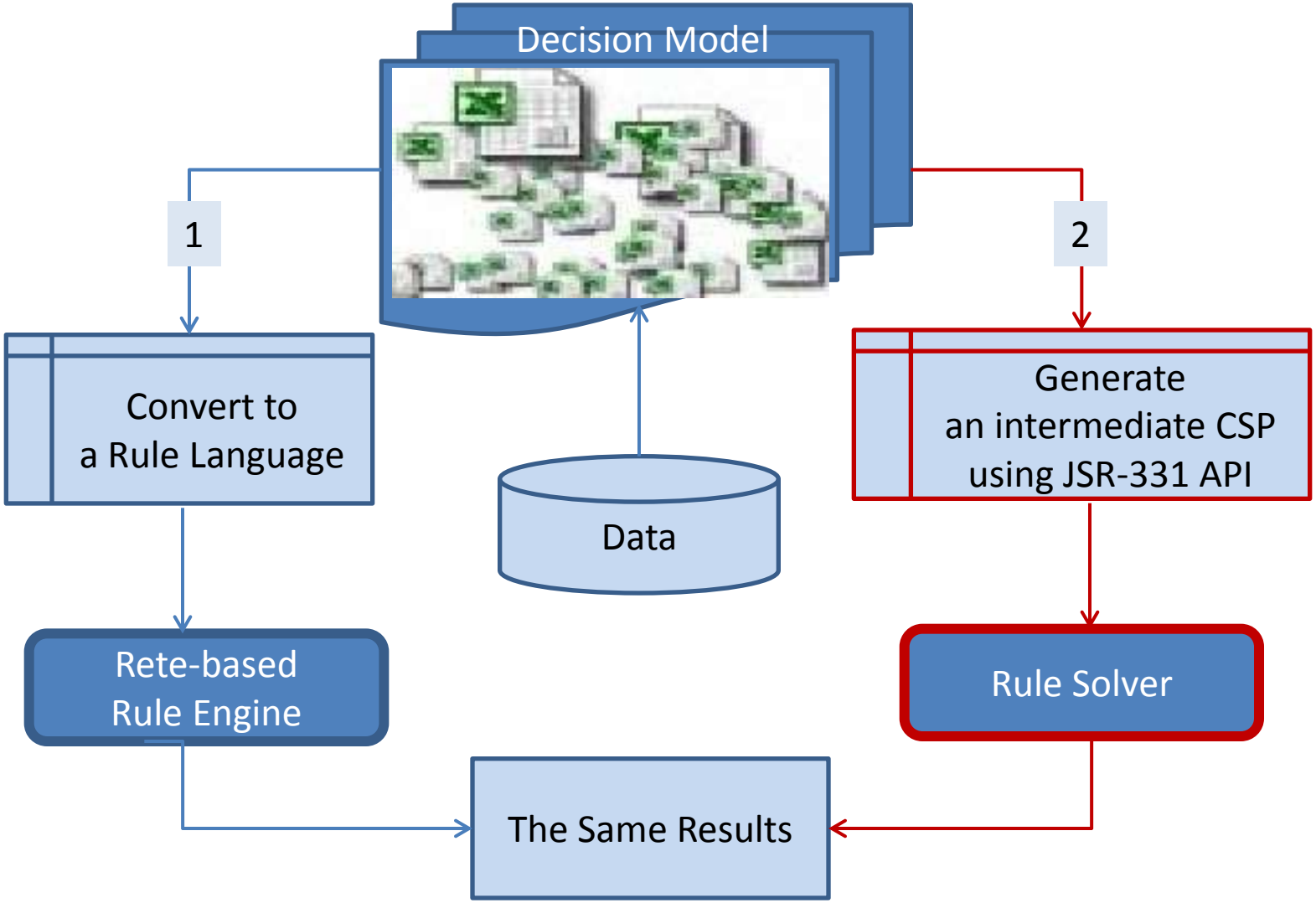
# OpenRules Approach

- 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

# New Rule Solver

- **A new Rule Solver can execute <u>inferential</u> 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 (the first alternative to Rete)

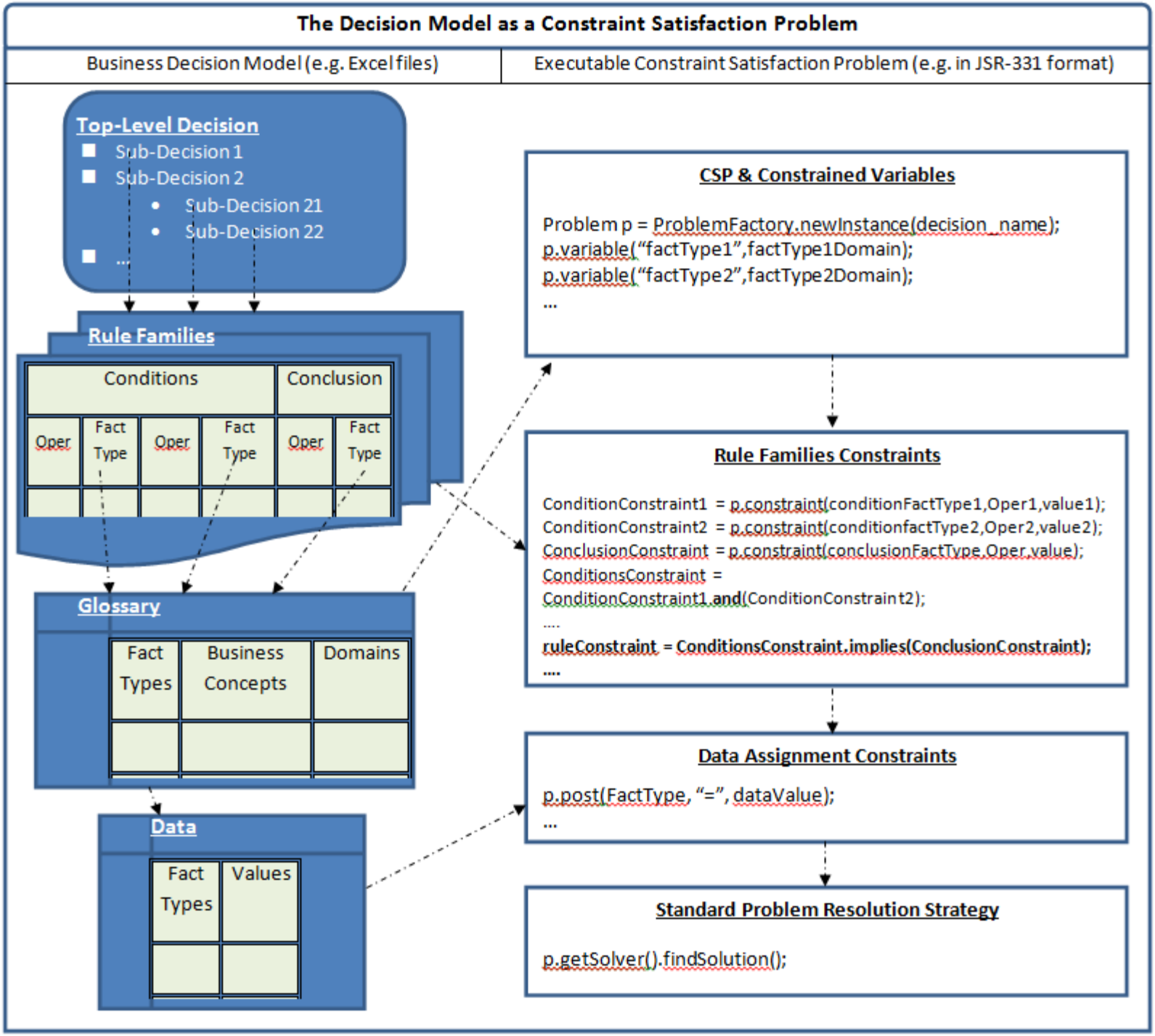
# Rule Solver Validates and Executes Decision Models



# 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

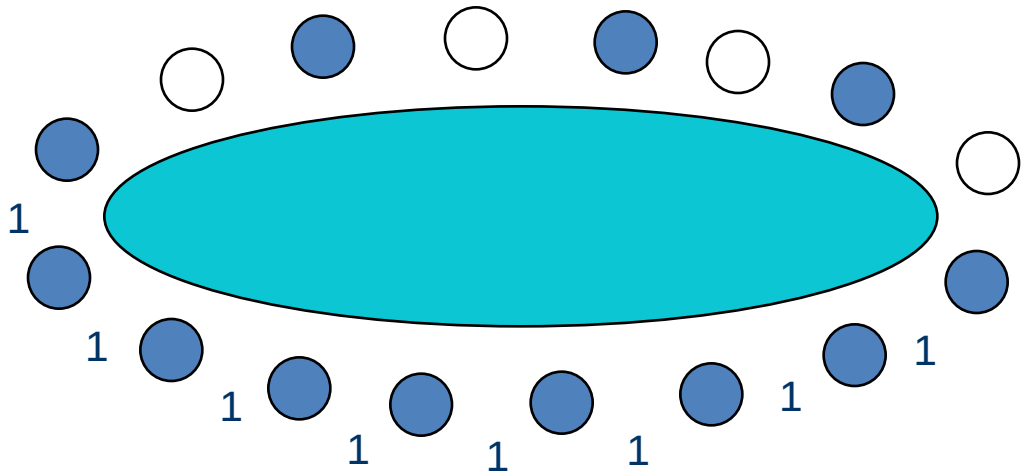


# Comparing BR and CP

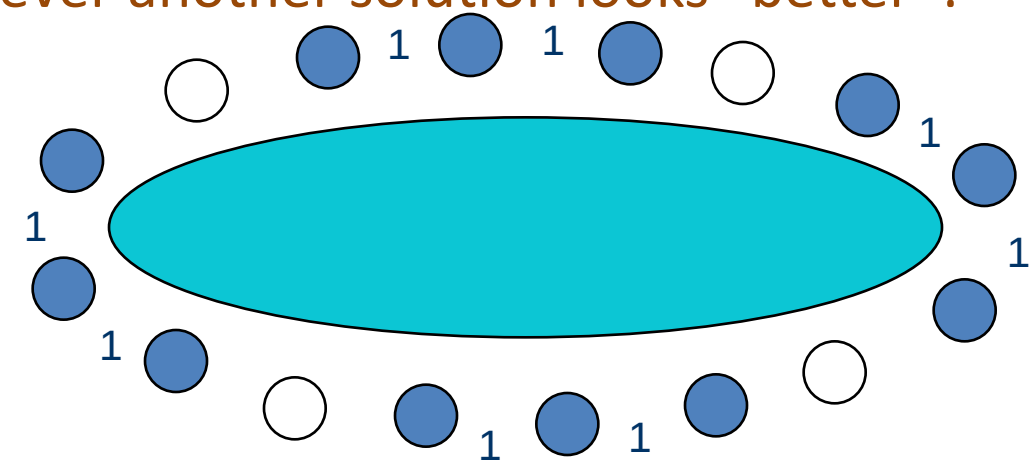
- BR Advantage:
  - Rules Repository is managed by business people while Constraint Store is usually under control of software developers
- CP Advantage:
  - Rules usually have to consider All (!) possible combinations of the problem parameters
  - Constraints do not have to cover all situations but 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

# Decision Model Validation

- Validate Rules Consistency using Constraint Propagation
- Checks Rules Completeness
- Validation can be done
  - Within One Rule Family
  - Across All Rule Families

# Summary

- Decision model can be used as a common input for Rete-based or Constraint-based rule engines
- Rule Solver can be used as a rule engine for execution of decision models defined by business users
- Rule Solver validates consistency and completeness of decision models
- Rules Solver goes beyond traditional BR problems