

Implementing the UServ Product Derby with the TAKE rule compiler

Jens Dietrich, Massey University NZ
j.b.dietrich@massey.ac.nz

Two versions

Version 1 – static compilation, no external fact sources used – can be started from TAKE web page (requires Java WebStart 1.6).

<http://code.google.com/p/take>

Subversion:

http://take.googlecode.com/svn/trunk/RuleML2007_challenge/

Version 2 – dynamic compilation, uses facts from lightweight web service and DB.

I.e., rules are compiled when loaded!

Subversion:

http://take.googlecode.com/svn/trunk/RuleML2007_challenge/

About the Scenario

- UServ Product Derby Case Study
- Car insurance rules
- published by the business rule forum to benchmark rule applications
- http://www.businessrulesforum.com/2005_Product_Derby.pdf
- 69 rules -> 111 source code files, 354 classes
- Compilation takes 2567 ms on a system with a 2.0GH T5600 dual core processor, 2.0GB of RAM running on Ubuntu 7.10 with Java 1.6.
- Handicaps: loggings switched on, code pretty printing

The Application

- Simple UI
- Changes in input area cause changes in output area – queries are reissued.

The screenshot shows the UServ application window. At the top, there are buttons for 'exit', 'show rules', 'load rules', and 'about'. Below these are three tabs: 'driver details', 'car details', and 'policy details'. The 'driver details' tab is active, showing various input fields and checkboxes. The 'output' section has two tabs: 'driver eligibility' and 'policy premium'. The 'driver eligibility' tab is active, showing a list of eligibility criteria with checkboxes and a summary box on the right. The summary box displays 'policy eligibility score' with values 100, 30, and a total of 130. At the bottom, there is a footer with the text 'UServ Product Derby Cas' and 'Forum (http://www.businessrulesforum.'

input	
id	424242
age	18
location	unknown
is married	<input type="checkbox"/>
is male	<input checked="" type="checkbox"/>
is elite	<input type="checkbox"/>
is preferred	<input type="checkbox"/>
has drivers training from school	<input type="checkbox"/>
has drivers training from licensed driver training company	<input type="checkbox"/>
has taken a senior citizen drivers refresher course	<input type="checkbox"/>
number of accidents involved in	0
number of moving violations in last two years	0
number of years with UServ	0

output	
driver category	young driver
is eligible	no
has training certification	no
is high risk driver	no
is long term client	no
policy eligibility score	100 30 total: 130
insurance eligibility	must be reviewed by underwriting

input area

query results

show explanations

Rule Implementation

- Now uses TAKE scripts, R2ML version planned.
- Comprehensive use of meta data in scripts.
- Meta data can be retrieved in application (use [?] buttons).

Access Resource from DB

- List of special locations is read from HSQL DB
- External fact store wraps a JDBC result set

@author=Jens Dietrich

@date=2007-10-18

@description=list of special locations

external DP_00x: **specialLocation**[Driver]

@category=Driver Premiums Rule Set

@author=Jens Dietrich

@date=2007-09-12

@description=If young driver and married and located in special location, then increase premium by \$700.

DP_01: if driverCategory[driver,"young driver"] and isMarried[driver] and **specialLocation**[driver] then **additionalDriverPremium**[driver,700]

Access Resources from WebService

- DUI conviction lookup implemented as web service
- dummy web service returns a boolean for an id (id contains 42 – must have a DUI conviction)
- RESTful – simple GET request, not SOAP

@author=Jens Dietrich

@date=2007-09-12

@description=If the driver has been convicted of a DUI, then the driver qualifies as a High Risk Driver.

DE_DRC01: if **hasBeenConvictedOfaDUI**[driver] then
isHighRiskDriver[driver]

@author=Jens Dietrich

@date=2007-10-19

@description=get info about DUI conviction from external data source

external DRCx: **hasBeenConvictedOfaDUI**[Driver]

Built in aggregation

- Functions can be defined using standard` aggregation functions
- similar to GROUP BY construct in SQL

aggregation eligibilityScore = **sum** x **policyEligibilityScore**[car,driver,x]

@author=Jens Dietrich

@date=2007-09-12

@description=If car is Provisional then increase policy eligibility score by 50.

ES_01b: if autoEligibility[car,"provisional"] then **policyEligibilityScore**[car,driver,50]

@author=Jens Dietrich

@date=2007-09-12

@description=If eligibility score is less that 100, then client is eligible for insurance.

ES_04: if **eligibilityScore**(car,client)<100 then insuranceEligibility[car,client,"eligible"]

Negation

- Negation is polymorphic
- In predicate is defined in object model, negation is explicit
- Otherwise, negation is negation as failure

explicit negation – predicate references Java property

@description=If the car has Driver and Passenger airbags then lower the premium by 15%.

AD_02: if hasAirbags[car] and hasFrontPassengerAirbag[car] and **not** hasSidePanelAirbags[car] then premiumDiscount[car,15]

NAF – negated predicate is defined by rules

@description=Driver is a Typical Driver is all of the following are true: Not a Young Driver, Not a Senior Driver.

DP_07: if **not** driverCategory[driver,"young driver"] and **not** driverCategory[driver,"senior driver"] then driverCategory[driver,"typical driver"]

Reloading Rules

- Will trigger recompilation.
- Generated rules are stored in `./takeWorkingDir/src`
- Compiled rules are stored in `./takeWorkingDir/bin`
- package name contains timestamp for versioning
- Rules loaded with `URLClassLoader` into application

Reloading Rules - code

```
KnowledgeBaseManager<UserRules> kbm =  
    new KnowledgeBaseManager<UserRules>();  
Bindings bindings = new SimpleBindings();  
bindings.put("HighTheftProbabilityAutoList",HighTheftProbabilityAutoList.getList()  
);  
bindings.put("CurrentYear",new GregorianCalendar().get(Calendar.YEAR));  
bindings.put("NextYear",new GregorianCalendar().get(Calendar.YEAR)+1);  
Bindings factStores = new SimpleBindings();  
factStores.put("DP_00x", new SpecialLocationsSource());  
factStores.put("DRCx", new DUIConvictionInfoSource());  
FileInputStream in = new FileInputStream("user.take");  
UserRules kb = kbm.getKnowledgeBase(  
    UserRules.class,  
    ScriptKnowledgeSource(in),  
    bindings,  
    factStores);
```