INTELLIGENT DECISION SUPPORT SYSTEMS
(Master in Artificial Intelligence, UPC-URV-UB)

Spring semester, Course 2013/2014
Miquel Sànchez Marrè, Karina Gibert
February, 2014

Practical Work 3 (PW3, Group work) due on:

The objective of this exercise is a research work, including some practical software development and data analysis, about several features of Intelligent Decision Support Systems.

There will be 4 different practical works. One work for each working group.

PW 3.1 - Enhancing a Rule Inference Engine integrated in an Intelligent Decision Support Systems Tool (GESCONDA) *(Advised by Miquel Sànchez-Marrè)*

Overview of the work
The work has two sub-goals. Firstly, the design and implementation of some functionalities in a rule inference engine, which has already been constructed in the GESCONDA software. Some research task must be undertaken to propose an efficient implementation to import facts from the current active database in GESCONDA to the Fact Base of the Rule-Based Reasoning System. In addition, the inference rules produced both from the Rule Induction module and from the Decision Tree module of GESCONDA could be imported into the Knowledge Base of the Rule-Based Reasoning System.
The second issue is to test the implemented software with a real case where the use of some inference rules will be needed for helping the decision support tasks involved in a nutritional problem, which will be provided to the students.

Constraints
The developed software will be integrated into an already existing system, named GESCONDA, which is coded in Java. The code developed must be in Java, and should be non-intrusive regarding the existing functionalities and data structures of existing GESCONDA code.

Material to deliver:

1. The new version of the software both in font files and jar
2. The new version of the user manual including your extension
3. A report explaining.

a. the intervention performed in GESCONDA from both methodological and computational point of view
b. the resulst of testing the application with some real scenarios

Already implemented functionalities

Design of the Rule-Based Reasoning System (RBRS)

- Fact Base
  - Definition of a new fact
  - Modification of a fact of the Fact Base
  - Removing a fact from the Fact Base
- Knowledge Base
  - Management of Modules
    - Definition of a new module of the KB
    - Modification of a module of the KB
    - Removing a module from the KB
  - Management of Rules
    - Definition of a new rule of the KB
    - Modification of a rule of the KB
    - Removing a rule from the KB
  - Management of Meta-rules
    - Definition of a new meta-rule of the KB
    - Modification of a meta-rule of the KB
    - Removing a meta-rule from the KB
- Inference Engine
  - Forward chaining: deductive reasoning from data to the goals
  - Backward chaining: validation of a concrete goal/s with the available data

Execution of the Rule-Based Reasoning System

- Interactive execution: using the rule engine, in an interactive way with the user to solve a single problem
- Batch execution: solving several problems, described with the available data in a file, simulating a process of the real world. Each entry in the file is the descriptive data of a new problem to solve. No interaction is allowed.

New desired functionalities

Design of the Rule-Based Reasoning System (RBRS)

- Fact Base
  - Importation of a fact from the active GESCONDA Data Base to the Fact Base
- Knowledge Base
  - Management of Rules

- Import rules from GESCONDA module of Classification Rules to the KB
- Import rules from GESCONDA module of Decision trees to the KB

Uncertainty modelling and management

- A mixture of a certainty factor model and a fuzzy model would be a reasonable modelling. This model will be explained to the students by the teacher.

Management of the Rule-Based Reasoning System (RBRS)

- Save a current RBRS
- Load an existing RBRS

Auto-explicative component

- Information about how a problem has been solved by the rule inference engine on some kind of console.

Rules
Rules will have only conjunctive propositions in the antecedent part.
The consequent will be only one proposition.

Example: Temperature1 > 17 & (Temperature1 < Temperature2) -> humidity = high

Meta-rules
The antecedent will have the same form of a rule.
The consequent are special propositions. Admitted consequents will be the following:

- act-rule <list of rule identifiers>
- deact-rule <list of rule identifiers>
- forward
- backward <list of facts to be validated>
- go-module <list of module identifiers>
- stop-module
- halt

Syntax of the Readable/Writable objects in the Rule-Based Reasoning Module

The different objects are defined as follows in EBNF notation, to be readable/writable for the GESCONDA system from/to external files:

<rule> ::= "rule" "[" "name" "=" id "," "norm" "=" "(" <antecedent> ")" "->"
<consequent> "]" "\n" ;

<antecedent> ::= <condition> | <condition> <logicoperator> <antecedent> ;

<condition> ::= "(" <term> <equalityoperator> <term> ")" ;

<term> ::= id | <value> ;

<equalityoperator> ::= "<" | ">" | "=" | "<>" | "<=" | ">=" ;

<logicoperator> ::= "&" | "|" ;

<numericaloperator> ::= + | - | * | / ;

<consequent> ::= id "=" <value> | id "=" <computation> | "message" "=" string |
"arule" "[" {id,} "]" | "drule" "[" {id,} "]" | "forward-mod" "[" {id,} "]" |
"backward-mod" "[" {id,} "]" | "track" "[" {id,} "]" ;

<value> ::= "value" "[" <number> "]" ;

<computation> ::= "compute" "[" <operation> "]" ;

<operation> ::= <number> | <number> <numericaloperator> <operation> |
"(" <operation> ")" ;

<number> ::= int | double ;

<fact> ::= "fact" "[" "name" "=" id "," "question" "=" <option> "," "asked?" "="
<option> "," "type" "=" <type> ","value" "=" <factvalue> "] "\n" ;

<opcion> ::= "true" | "false" ;

<factvalue> ::= string | <number> | <enumeratedvalue> ;

<enumeratedvalue> ::= "enumerated" "[" "value" "=" string "," "type" "=" "ORDERED" |
"NOT ORDERED"," "values" "=" "[" string <enumeratedvalue> "]" "]"

<module> ::= "module" "[" "name" "=" id "," "forward" "=" <opcion> "," "rules" "="
"[" string <chainnames> "]" "," "metarules" "=" "[" string
<chainnames> "]" "," "children" "=" "[" <childrenlist> "]" "]"

< childrenlist > ::= "null" | string < chainnames>

<chainnames> ::= "" | "," string < chainnames>

<u>Example of a fact:</u>

fact [name=price, question="How much it costs?", asked?=true, type=numerical, value=2]

Example of a rule:

rule [name = rule3, norm = ((espera = value[0.5]) & (coche < sal)) -> message = "executed rule3"]

Example of a meta rule:

rule [name = metarule2, norm = ((velocidad > value[120]) & (tipovia = value[autopista])) -> arule[rule1,rule3]

rule [name = talcual4, norm = (coche < sal) -> drule[rule2]]

rule [name = talcual5, norm = (avance > retroceso) -> forward-mod[m1,m2,m3]]

Example of a module:

module [name = mod1, forward = true, rules = [rule1, rule2, rule3], metarules = [metarule1,metarule2], childrenlist = null]