isec
Politécnico de Coimbra

Knowledge and Reasoning

Licenciatura em Engenharia Informática: 2º ano - 2º semestre 2022/2023

Practical Class nº 6

Expert Systems: Health diagnosis example

1. Implementing an Expert System using Drools – diagnosis v. 1

In this exercise, we will implement an Expert System that, given a set of symptoms associated with a set of patients, is able to identify the correct diagnosis.

1.1 Creating a Drools Project

- Start Eclipse and do: **File - New Project - Drools Project**
- Name the project (Diagnosis1)

1.2 Create two Java Classes:

- **Class Name:** Patient
- **Fields:** id (String), name (String), age (int), diagnosis (String)
- Add the builder; getters and setters

- **Class Name:** Symptom
- **Fields:** patientID (String), symptom (String)
- Add the builder; getters and setters

The two classes are linked through the fields ***id*** and ***patientID***. This way we will know which symptoms belong to a specific patient.

1.3 Insert facts into the working memory

In the DroolsTest.java function, insert the following facts. The **diagnosis** field is empty and will be filled in by the rules when the appropriate diagnosis is found.

```
Patient P1: ("001", "Ana Melo", 12, "");
Patient P2: ("002", "Rui Costa", 13, "");
Patient P3: ("003", "Joana Martins", 85, "");
Patient P4: ("004", "Pedro Torres", 53, "");
Patient P5: ("005", "Ana Gomes", 93, "");

Symptom S1: ("001", "fever");
Symptom S2: ("001", "spots");
Symptom S3: ("002", "fever");
Symptom S4: ("003", "fever");
Symptom S5: ("003", "pain");
Symptom S6: ("004", "fever");
Symptom S7: ("004", "pain");
Symptom S8: ("004", "spots");
```

From this information, we can see that symptoms S1 and S2 belong to patient P1, symptom S3 to patient P2, symptoms S4 and S5 to patient P3 and symptoms S6, S7 and S8 to Patient P4. Patient P5 has no associated symptoms.

1.4 Implement the following rules

a) Rule 1: "rash 1"

If a particular patient aged 15 or more has a fever, spots and pain, update his diagnosis (**diagnosis** attribute) to "rash 1" using the "setDiagnosis" method. Also print the diagnosis on the RHS (then) component of the rule.
>> *Patient <name> was diagnosed with Rash 1.*

Test this rule: it should work well. Output:

>> Patient Pedro Torres was diagnosed with Rash 1

Now, also update the new diagnosis in Drools working memory, using: ***update ($p);*** where the variable **$p** is the variable you used to load instances of the ***Patient*** class

Test this rule: now you should check that the system goes into a "loop", successively printing *"Patient Pedro Torres was diagnosed with Rash 1".*
This is due to the fact that the ***update*** instruction updates the facts in memory and informs Drools that, therefore, the rules must be reconsidered. In other words: after the update, this rule becomes active again, and fires again. How to overcome this problem?

One of the hypotheses is to test, in the LHS of the rule (antecedent) if the diagnosis is already known or not. To do this, add a test to the diagnostic value in the LHS of the rule, such as && **diagnosis** == ””. Test the rule again. The loop is now removed.

b) Following an analogous process, create the following rules:

Rule 2: “Rash 2”
If a particular patient under the age of 15 has a **fever** and **spots** update and print his diagnosis for ***"rash 2"***

Rule 3: “flu”
If a certain patient has a **fever** and **pain**, update and print his diagnosis as "***flu***"

Rule 4: “cold”
If a certain patient only has a ***fever***, update and print his diagnosis as a “**cold**”

c)

Rule 5: “no symptoms”
“has no symptoms”
If there are no symptoms in the working memory for a given patient, update and print the diagnosis as “***no symptoms***”

To test this condition, you must use the ***forall*** command, similar to the following

```
 $p:Pacient (diagnosis == “”)
forall($s:Symptom(idPacient!=$p.getId()))
```

Test the expert system, you should get the following output:

```
Patient Pedro Torres was diagnosed with Rash 1
Patient Ana Melo was diagnosed with Rash 2
Patient Joana Martins was diagnosed with Flu
Patient Rui Costa was diagnosed with a Cold
Patient Ana Gomes has no associated symptoms
```

d)
Note that the rules created must be triggered in the order in which they were created, that is, with higher priority for the most restrictive ones: patients who show more symptoms must be diagnosed first, otherwise, for example, “fever” + “pain” + "spots" could trigger the "cold" rule (which only requires "fever") and after that - because the diagnosis would be different from "" - it could never be updated to the correct diagnosis, "rash 1".

These situations occur whenever the coverage of the rules overlaps, that is, when one or some rules trigger with examples that also trigger other, more restrictive ones. To avoid this, each rule has to be given priority, using the **salience** command. The highest priority is indicated by **salience 100**, and the lowest by **salience 0**.

Assign the appropriate priorities to the rules created. Try changing their order and test the program. The results must always be the same, and correct.

e) Create one more rule to print the ID, name and diagnosis of each patient. Naturally, this rule should only be triggered after all the previous ones.
Comment all print commands performed by the previous rules. Test the program. It should work correctly.

2. Implementing an Expert System using Drools – diagnosis v. 2

We will now solve the problem the loop in the rules, focused on the previous section, in another way. The idea is to create a new class in which the associations between patients and their diagnoses are inserted, and at the end to print the contents of this class. In this way, it is possible:

1) avoid updating the Patient class
2) removing patients from the memory of facts as they are diagnosed, thus avoiding the triggering of lower priority and less restrictive rules

Note: To remove facts, use the ***retract*** command.

a) Create a new project named Diagnostico_V2. Create 3 Classes now:

- **Class Name:** Patient
- **Fields:** id (String), name (String), age (int)
- Add the builder; getters and setters

- **Class Name:** Symptom
- **Fields:** patientID (String), symptom (String)
- Add the builder; getters and setters

- **Nome da Classe:** Diagnosis
- Campos: idPacient (String), namePaciente (String), descriptiom (String);
- Add the builder; getters and setters

b) Rewrite the rules, changing the following:

i) Remove the diagnostic test == ”” in the LHS condition of the rules
ii) Remove the instructions from the RHS component of the rules:

```
$p.setDiagnostico("...");
update($p);
```

i) In the RHS (conclusion of the rule) add the respective diagnosis to the Diagnostic class. To do this, use the insert command, similar to the following:

```
insert (new Diagnosis ($p.getId(), $p.getName(), “Rash 1”));
```

where $ p is the variable used to obtain the LHS patient list for the rule;

ii) In each rule, and after this insert, remove patients already diagnosed from the list of facts, using the retract command, in a similar way to the following:

```
            retract ($p);
```

where $p is the variable used to obtain the LHS patient list for the rule;

iii) Modify the print rule to now use the **Diagnosis** class (instead of Patient) to print the results.

iv)

v) Insert in the working memory the same facts as the previous example:

```
Pacient p1 = new Pacient("001","Ana Melo",12);
Pacient p2 = new Pacient("002","Rui Costa",13);
Pacient p3 = new Pacient("003","Joana Martins",85);
Pacient p4 = new Pacient("004","Pedro Torres",53);
Pacient p5 = new Pacient("005","Ana Gomes",93);

Symptom s1 = new Symptom("001", "fever");
Symptom s2 = new Symptom("001", "spots");
Symptom s3 = new Symptom("002", "fever");
Symptom s4 = new Symptom("003", "fever");
Symptom s5 = new Symptom("003", "pain");
Symptom s6 = new Symptom("004", "fever");
Symptom s7 = new Symptom("004", "pain");
Symptom s8 = new Symptom("004", "spots");
kSession.insert(p1);
....
```

vi) Test the program. It should work correctly.

```
Patient with id 005 Ana Gomes with diagnosis: no symptoms defined
Patient with id 002 Rui Costa with diagnosis: Cold
Patient with id 003 Joana Martins with diagnosis: Flu
Patient with id 001 Ana Melo with diagnosis: Rash 2
Patient with id 004 Pedro Torres with diagnosis: Rash 1
```

vii) Test the program using more patients and facts. For that, also use a rule to insert them in the memory of facts, similar to the following:

```
rule "Insert more facts"
salience 110
when
then
insert (new Patient ("006", "Carlos Lopes", 22));
      insert (new Patient ("007", "Maria Ferreira", 25));
      insert (new Patient ("008", "Nuno Gomes", 60));
      insert (new Patient ("009", "Julio Lopes", 58));
      insert (new Patient ("010", "Rui Saraiva", 45));
      insert (new Symptom ("006", "fever"));
      insert (new Symptom ("006", "pain"));
      insert (new Symptom ("006", "spots"));
      insert (new Symptom ("007", "fever"));
      insert (new Symptom ("008", "pain"));
      insert (new Symptom ("010", "spots"));
end
```

viii) In the inserts of vii) note that patients 008 and 010 have symptoms that do not correspond to any disease. One possibility of contemplating this situation is to add the rule

```
rule "with symptoms but do not correspond to any diagnosis"
salience 10
when
      $p1:Pacient()
then
      insert   (new   Diagnosis   ($p1.getId(),   $p1.getName(),   "With
symptoms, but do not correspond to any diagnosis"));
      retract ($p1)
end
```

It should cover only the patients who remain after they have all been removed through the various retracts of the previous rules, and therefore their salience must be lower than that of the other diagnostic rules (in the example, salience = 10, remaining rules salience = 50 or more)

Including this rule, the results should be:

```
• Patient with id 008 Nuno Gomes with diagnosis: With symptoms, but not
  corresponding to any diagnosis
• Patient with id 010 Rui Saraiva with diagnosis: With symptoms, but not
  corresponding to any diagnosis
• Patient with id 009 Julio Lopes with diagnosis: no symptoms defined
• Patient with id 005 Ana Gomes with diagnosis: no symptoms defined
• Patient with id 007 Maria Ferreira with diagnosis: Cold
• Patient with id 002 Rui Costa with diagnosis: Cold
• Patient with id 003 Joana Martins with diagnosis: Flu
• Patient with id 001 Ana Melo with diagnosis: Rash 2
• Patient with id 006 Carlos Lopes with diagnosis: Rash 1
• Patient with id 004 Pedro Torres with diagnosis: Rash 1
```

3. Implementing an Expert System using Drools – diagnosis v. 3

We are now going to implement the final version of the program, with the possibility to collect facts indicated by the user, and then perform the diagnosis of a single patient. For this:

a) Create a new project named Diagnosis_v3, containing the 3 classes used in the previous version.

b) Keep the diagnostic rules used in the previous version, except for the "insert more facts" rule, created in B-vii), which must be eliminated.

c) To collect symptoms given by the user, use a rule with syntaxes of the type:

```
rule "Input data "
salience 110
when

then
	System.out.println("Patient's Name?");
	Scanner input1 = new Scanner(System.in);
	String Name = input1.nextLine();
	String r1;

	System.out.println("Age?");
	Scanner input2 = new Scanner(System.in);
	int Age = input2.nextInt();

	insert(new Patient("001", Name, Age));

	System.out.println("Are you in pain?");
	r1 = input1.nextLine();

	if (r1.equals("S")) {
		insert(new Symptom("001", "pains"));
	}

	System.out.println("Are there stains?");
	r1 = input1.nextLine();

	... // TO COMPLETE
end
```

Add the instruction: `import java.util.Scanner;`

Tip: After creating this rule, you can create another one for debugging purposes, intended only to print all the known facts and verify that they were correctly inserted in the Drools working memory.

d) In the **Drools.Test.Java** application, remove all facts inserts, keeping only the activation of the rules, kSession.fireAllRules();

e) Test the program, not forgetting the situations in which the combination of symptoms does not correspond to any diagnosis, and in which the patient has no symptoms.