Assignment 2 Rule-Based Systems

Development of Knowledge Systems,
September 2009

Getting started

Read: Chapter 1 and 2 of the CLIPS User's guide.
Hand in: Nothing.

Run CLIPS and perform the following steps:

a) Devise a proper representation for the fact that the weather is rainy.
b) Add the fact to the CLIPS fact-list.
c) Check if the fact has been added to the fact-list.
d) Devise a proper representation for the rule that when the weather is rainy, the streets get wet.
e) Add the rule to the CLIPS rule-base.
f) Check if the rule has been added to the CLIPS rule-base.
g) Check the facts; did the facts change? Give an explanation for this.
h) Check the agenda. Give an explanation for what's on the agenda.
i) Run the program.
j) Check the facts; did the facts change? Give an explanation for this.
k) Check the agenda; did the agenda change? Give an explanation for this.
l) Reset the facts. Check if the facts have been reset.
m) Reset the rules. Check if the rules have been reset.

Above you used a runtime knowledge base, which only exists in the memory of your computer (and is "destroyed" when you close CLIPS). To create a "permanent" knowledge base you have to create a CLIPS file. You can create a CLIPS file by opening a text editor (such as notepad) and save the file using the ".clp" extension. You can load this file into CLIPS by choosing File|Load in the main menu. Do not forget to type (reset) each time before you run the program!

Hand-in Exercises

Read: CLIPS User's guide;
Useful introductions to CLIPS: www.ida.liu.se/~TDDB66/labkomp/CLIPSintro.pdf
Hand in: A report with the answers to the exercises (the exercises specify what needs to be added to the report), and the .clp files of exercises 1, 2, and 3.

1. Programming in CLIPS

Represent the following facts and rules in CLIPS (in a general manner).

> All animals have skin. Fish is one kind of animal, birds are another type and mammals are a third kind. Normally fish has gills and can swim, while birds have wings and can fly. While fish and birds usually lay eggs, mammals do not. Although sharks are fish, they do not lay eggs. They are very dangerous. Salmon is another fish and is considered a delicacy. Canary is a bird and is yellow. Ostrich is a bird, which is very tall, but cannot fly, only walk.

Using your CLIPS program answer the following questions

1. Can canaries fly?
2. What is the color of canaries?
3. Can ostriches fly?
4. Do canaries have skin?
5. Are sharks dangerous?
6. Can you eat salmon?

You can use deffacts and deftemplate statements to describe more general relationships.
Hand in: the knowledge base (.clp file) and add to your report a short readme about how to test your program to get the correct answers to the questions above. Also include a trace of the queries used in CLIPS to determine the answers.

2. Rule-based systems modeling and programming
Consider the following proposal for a "living expenses contribution" (LEC).

LEC is only available for poor people. Poor people are people without a job or people with debts. They have difficulties to pay for their accommodation and food. People with more than 100.000 Euro earnings per year or with more than 50.000 Euro on their savings account are rich. Young people (less than 25 years old) with rich parents do not get any LEC; they should be supported by their parents.

a) Put the information about LEC in a decision table. Make sure that the structure of the knowledge is reflected in your diagram. It is OK to leave out some aspects that are irrelevant with respect to the LEC outcome, just make sure that all important things are there.

b) Implement your program in CLIPS, such that, after asserting all the facts (conditions), the program adds the correct output value to the factbase; either they do get LEC, or they do not get LEC.

Hand in: Your report should include the decision tables from **a)** and a short description of the rules of your CLIPS program. Please attach the CLIPS file (.clp) to your hand in as well.

3. Programming in CLIPS

a) Family knowledge base.

The purpose of this exercise is to create a generic knowledge base that is easily extended and maintained. An interactive shell can be created for those that have time to do so, but is not mandatory (see the example on the website for how to create interactive knowledge bases in CLIPS).

Construct a knowledge base for extended families.

1. Define a generic (short) manner of representing people, such that your knowledge base can derive the following relationships and properties (via generic rules): gender (male, female), age, mother, father, sister, brother, husband, wife.
2. Populate the knowledge base with at least 10 individuals. Each individual must have at least one relationship and in average 2 properties.
3. Add the generic rules to make the system able to derive the relations and properties specified in 1.
4. The program must be able to (not necessarily interactive[1]):

i. add additional (new) members to the family;
ii. specify or change their relationships and properties;
iii. answer queries about properties (e.g., what is John's age?).
iv. list all properties about entities (e.g., what do we know about John's mother?);
v. identify individuals with a certain property (e.g., who is older than John?);
vi. answer queries about relationships (e.g., who are John's brothers?);

Hand in: Your report should include a diagram of the family tree you are using (the individuals and their relationships), a short description of your chosen representation (what do you store and why?), and a short readme about how to test your program to get the correct answers. Also, remember to hand in your knowledge base (.clp file).

b) Family relationships

Using the relationships defined in a), extend your database with the following relationships:

1. Parent
2. Sibling
3. Cousin
4. In-law
5. Grandfather, grandmother, grandparent
6. Uncle, aunt

Hand in: the knowledge base containing the specification of these relationships (.clp file). Your report should contain a (short) description of the added rules.

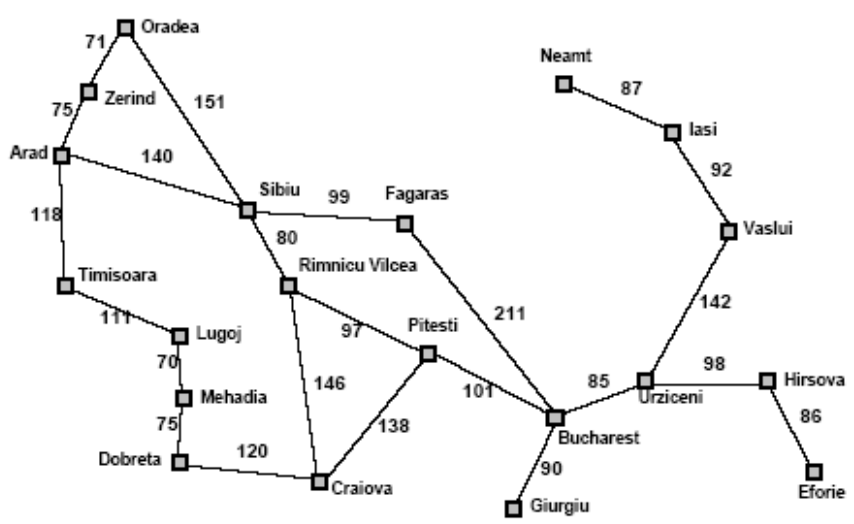
---

[1] Your program can be made to be interactive like the program on the website, but for this exercise it suffices if your program is generic enough that new knowledge can be added easily. If this option is chosen, your readme should reflect this by providing the details on how to add new knowledge to the knowledge base (e.g., "type assert(person(piet,50,mary)) to add piet, of age 50, to the knowledge base).

4. Search

Note: this exercise does not need to be programmed in CLIPS

This exercise concerns the route-finding problem using a map of Romania.



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 1: Example taken from Russel & Norvig: *Artificial Intelligence: A Modern Approach*

Consider the route-finding problem (from Arad to Bucharest) as a state space search problem.

a) Enumerate the states in the search space for this problem. How many distinct states are there?

b) Using each state as a node, and the travel actions as an arc show the resulting search tree. Simplify the state space, so you only travel in one direction (that is do not go back to states where you have been before) and only consider states that represent a choice point (more than one travel direction possible). When there is a choice of nodes, use alphabetical order, left to right.

c) Which is the order of traveled nodes for each of the following algorithms when searching for a (shortest) path between Arad and Bucharest?

(i) Dept-first search

(ii) Breadth-first search

(iii) A* search with straight-line distance heuristic

d) Which algorithm is most suitable in practice for solving route-finding problems such as this? Argument your answer by describing the properties of the different algorithms

Hand in: Your report should include the answers to all questions and supporting diagrams.