



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

DSIC
DEPARTAMENTO DE SISTEMAS
INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática



Dpto. Sistemas Informáticos y Computación
Escuela Técnica Superior de Ingeniería Informática
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

INTELLIGENT SYSTEMS

LAB ASSIGNMENT 1

DESIGN, IMPLEMENTATION AND EVALUATION OF A RULE-BASED SYSTEM

Sokoban game




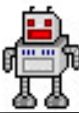



October 2018

Sokoban

Sokoban is a type of puzzle created in Japan which means “warehouse”. The goal of the game is that a player pushes all the boxes distributed around a grid in the warehouses.

In order to explain the rules and restrictions of the game, we will use the grid shown in the figure below as an example. Nevertheless, the student must design and write a CLIPS program that **works for any configuration of the grid**.

The grid of the figure displays grey cells that represent obstacles, three boxes, three warehouses and a robot (player). The grid cells are referenced by two numbers that indicate the row and column of the cell, respectively. Thus, in the figure, the robot is in position (4,1), there is a warehouse at (1,7), a box in cell (4,3), an obstacle at (3,5), etc.

	1	2	3	4	5	6	7	8
1								
2								
3								
4								
5								

The robot can move to an adjacent cell in any of the four following directions: up, down, right and left. Likewise, the robot can push a box to the upper adjacent cell, to the below cell, to the cell on the left or to the cell on the right of the box.

The robot can move to a cell in any of the four directions if:

- The destination cell does neither have an obstacle nor a box
- The destination cell is not a warehouse

For pushing a box the robot must be located in a cell adjacent to the box. The robot can only push the box to a cell that neither has an obstacle nor a box, or to a warehouse. For example, given the box in position (4,3) in the grid of the figure, the robot:

- can go to cell (5,3) and push the box upwards; the effect of this operation is that both the robot and the box are moved one row up; that is, the robot will end in cell (4,3) and the box in cell (3,3)
- can go to cell (3,3) and push the box downwards; the effect of this operation is that both the robot and the box are moved one row down
- cannot go to the cell on the right of the box because there is an obstacle in such cell
- can go to the cell on the left of the box but then it does not get to push the box because there is an obstacle in the cell on the right of the box

Warehouses have capacity for only one box so that when the robot pushes a box the warehouse becomes full and no more boxes can be stored.

The work to do consists in solving this version of the Sokoban game, using a state-based representation of the problem, with a RBS implemented in CLIPS. The BREADTH and DEPTH search strategies will be used to run the CLIPS program and solve the problem:

1. The program will request the user the maximum depth level of the tree to develop (see example of the 8-puzzle problem)
2. The program will return the depth level and number of generated nodes of the solution found for each execution of the RBS with the search strategies (no need to return the solution path)
3. Use a generic knowledge representation that is easily modifiable. The program should allow changing the grid dimensions, including new obstacles, boxes or warehouses without having to modify the rules.
4. All the initial information can be directly written within the `def facts` command.

GUIDELINES

1. The pattern that will represent the information of a problem state should comprise only the dynamic and modifiable information like the position of the robot, the position of the boxes and the contents of the warehouses (empty or full).
2. The static information of the problem (information that is unchangeable along the execution of the RBS) like the dimension of the grid or the position of the obstacles can be represented with patterns other than the pattern of the problem state.
3. There are two alternatives to encode the obstacles of the grid:
 - a. To explicitly **represent the cells that contain an obstacle**. In this case:
 - Since the information is static, a pattern different from the pattern that represents the problem state can be used
 - The rules to move the robot and push a box must check **THERE IS NOT AN OBSTACLE** in the destination cell, which implies using negated patterns (see the note below)
 - b. To explicitly **represent the cells that do not have an obstacle** (regardless the cell contains a box or a warehouse). In this case:

- Since the information is also static (a cell that initially does not contain an obstacle, it will never do), a pattern different from the pattern that represents the problem state can be used
- The rules to move the robot and push the box must ensure that the destination cell **DOES NOT HAVE an obstacle** and so this restriction can be checked using a positive pattern

NOTE

The CLIPS syntax does not allow to use `test` for evaluating a variable instantiated in a **negated pattern**. For example, let's assume we want to check that *there does not exist* a fact 'item' which value is twice as much as any element comprised in a fact 'list'. **IT IS NOT POSSIBLE** to write a CLIPS rule like:

```
(defrule R1
  (list $? ?n1 $?)
  (not (item ?x))
  (test (= ?x (* ?n1 2)))
  ...)
```

The rule must be instead written as:

```
(defrule R1
  (list $? ?n1 $?)
  (not (item =(* ?n1 2)))
  ...)
```

A function can be used on the LHS of a rule if an equal sign, `=`, is used to tell CLIPS to evaluate the following expression rather than use it literally for pattern matching.

EVALUATION OF THE LAB ASSIGNMENT

The assessment of the lab assignment will be through an **INDIVIDUAL** test that will take place on Wednesday October 24. Two turns, each of 45 minutes, will be arranged on October 24: first turn will be from 8.00 to 8.45 and second turn from 8.45 to 9.30.

The test will include some questions about the implementation, evaluation of the RBS or modifications of the code to address a new functionality in the Sokoban game.

A task in PoliformaT will be created for the lab assessment. The CLIPS program that solves the Sokoban game described above must be uploaded to the PoliformaT task (CLIPS program before the exam) as well as the new CLIPS code that tackles the modifications requested in the exam (CLIPS program after the exam) (**IMPORTANT:** indicate clearly the name of the two group members -if this is case- in the CLIPS file that contains the program that solves the problem presented in this document).