Planning and Approximate Reasoning

(MIA) Practical Exercise 1: Planner

Blocks world with 2 arms

- The deadline for the delivery of this exercise is **November 1st, 2017**
- There is a second submission date set to **January 28th, 2018**. The exercise will be extended with additional requirements.
- A zip file with the **source code** must be delivered.
- A detailed **documentation in PDF** is also required (see details below)
- The submission of the source and documentation must be done using Moodle.
- This exercise must be solved in groups of two people.

This practical exercise consists in the design and implementation of **a non-linear planner with goal regression**. The domain of application is an extension of the "blocks world", with some additional features:

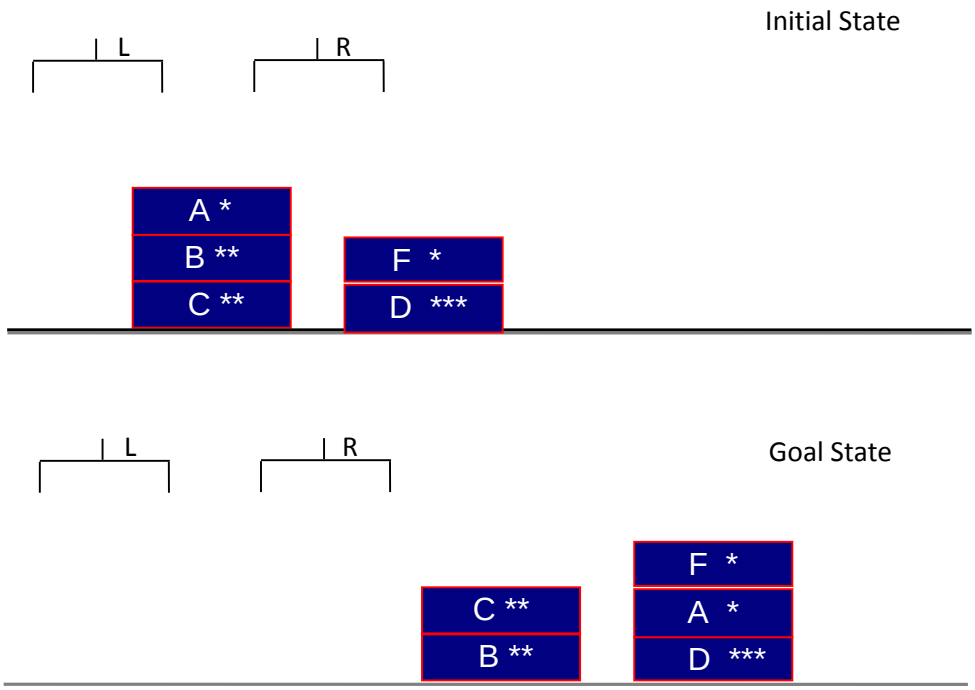
- In the table we can just have **a limited number of stacks of blocks**. That fact should be checked with an additional predicate, which should be updated after the application of some operators.
- We can only stack a block *x* over a block *y* if the second block is **heavier or has the same weight** than the first. We will consider only 4 possible weights: 1, 2, 3 and 4 kg.
- We have **two arms** (like a humanoid robot). While the right arm can pick up any block, the left one can only pick up light blocks of 1 kg.

Predicates:

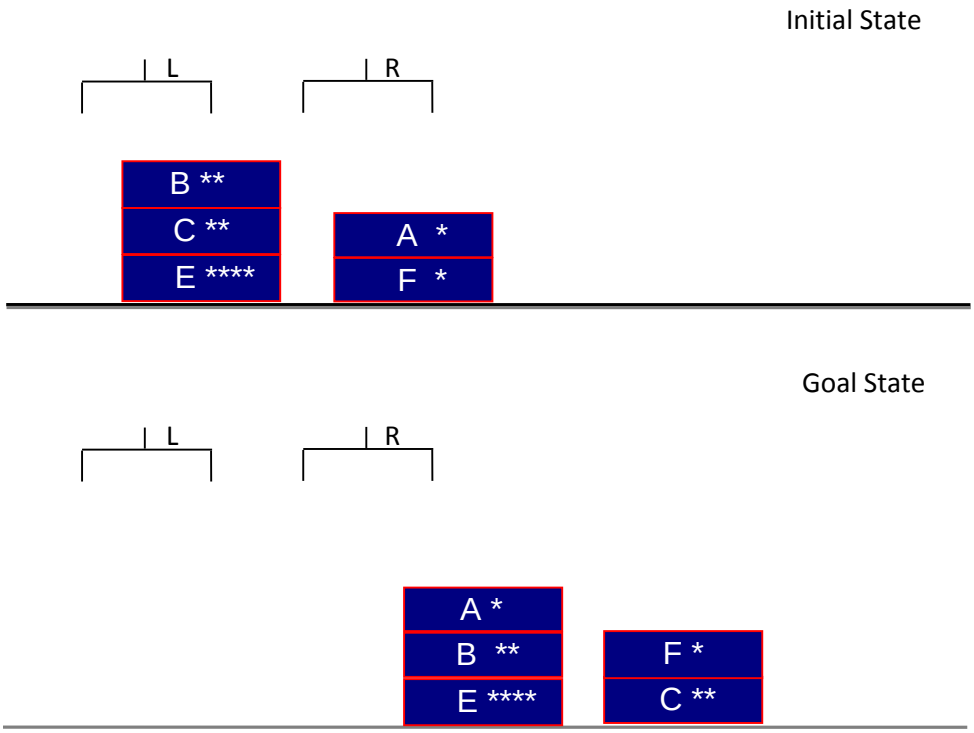
- **On-table**(*x*): *x* is placed on the table
- **On**(*x*,*y*): *x* is placed on *y*
- **Clear**(*x*): *x* does not have any object on it
- **Empty-arm**(*a*): the robotic arm *a* is not holding any object
- **Holding**(*x*,*a*): the object *x* is being held by the robotic arm *a*
- **Used-cols-num**(*n*): *n* block columns are being used
- **Heavier**(*x*,*y*): the object *x* weights more or the same than *y*
- **Light-block**(*x*): the weight of the object x is 1 kg

Example:
Now we present two examples, taking into account that the points in the boxes represent units of weight:

Problem 1. **(with MaxColumns=3)**



***Problem 2* (with MaxColumns=3)**



You have to:

- Define the **operators** properly.
- Implement the algorithm studied in this course, known as "**non-linear planner with goal regression"** in Java or Matlab to solve the problems presented above. It is important to choose an appropriate internal representation to manage the preconditions, to check the applicability of the operators, etc. Code should be readable (add comments).
- In order to improve the algorithm efficiency, design **different intelligent tactics** to aid the planning process during the detection of impossible states, the instantiation of variables, and other steps you consider.
- Input/output to the program will be done with text files with the following format (assume that the input file will not have any errors):

Input file:

```
MaxColumns=3
Blocks=A*,B**,C**,D***, F*;
InitialState=ON-TABLE(C),ON(B,C),ON(A,B),CLEAR(A);ON-TABLE(D),ON(F,D),
CLEAR(D),EMPTY-ARM(L),EMPTY-ARM(R);
GoalState=ON-TABLE(B),ON(C,B),CLEAR(C),ON-TABLE(D),ON(A,D),ON(F,A),CLEAR(R),
EMTPY-ARM(L),EMTPY-ARM(R);
```

Output file:

```
nn                    //  number of operators of the plan
ii                    //  number of states generated to solve the problem
op1, op2, op3, …   //  plan defined as a sequence of operators from initial to final state
-------------
// details of the states that were cancelled (not continued) reason with format:
p1,p2,p3 …   //predicates that define the state
repeated state, contradictory predicates, …  // reason for cancelling the exploration
-------------     //a line will separate each state
```

- Test your code with an extensive **set of test cases** including, at least, the two proposed problems and three more of your own. Discuss the solutions your program found and if there may or may not exist more optimal plans.
- Make a **graphical analysis** of the time and number of operators required in the different tests, depending on the values of the problem (for example, using different columns in the table, or with different number of blocks, etc).
- **Discuss** the limitations of your planner (f.i. show problems without solution, etc).

Documentation content:

1. Introduction to the problem
2. Analysis and formalization of the problem (operators, pre-conditions, special situations, etc.)
3. Planning algorithm (explaining the domain knowledge used to define the strategies or heuristics in the different steps of the method).
4. Implementation design (class diagram and details of the methods you consider more relevant).
5. Testing cases and results (show the contents of the stack during the execution, not only the final path). Analysis of the results (graphics with complexity, number of steps, etc.).
6. Instructions to execute the program.

Evaluation criteria:

30% analysis (formalization of the problem and discussion of the results obtained)
30% algorithm details and strategies for improving the planner
20% implementation
20% execution of the test suite and additional tests