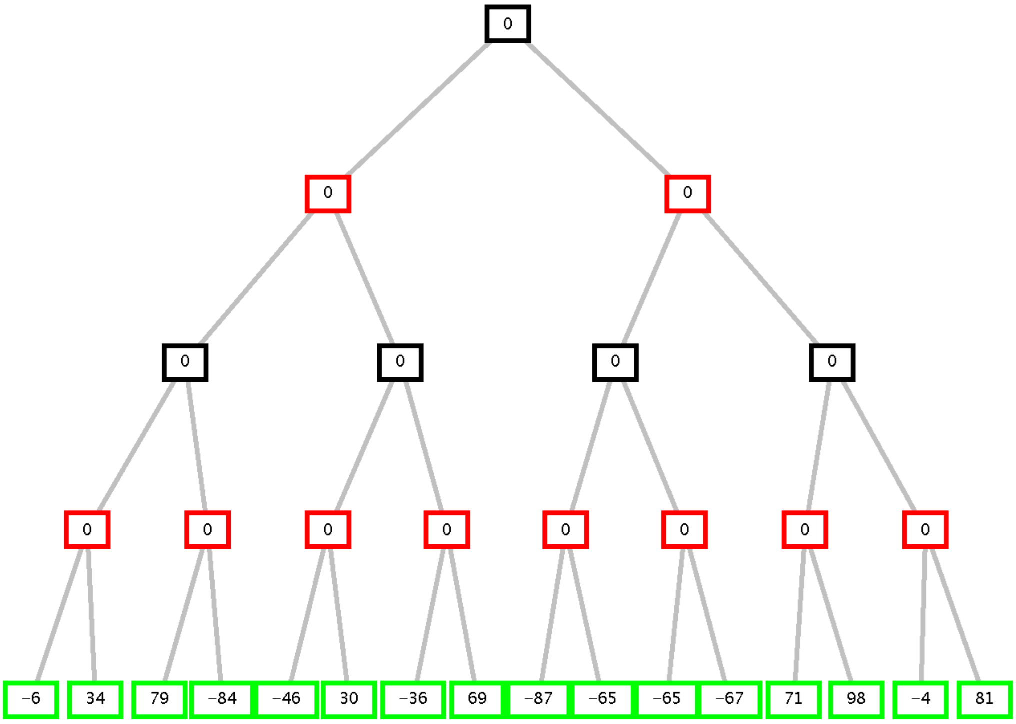
EXAM OF FUNDAMENTALS OF AI – FIRST MODULE
11/09/2020
PROF. MICHELA MILANO

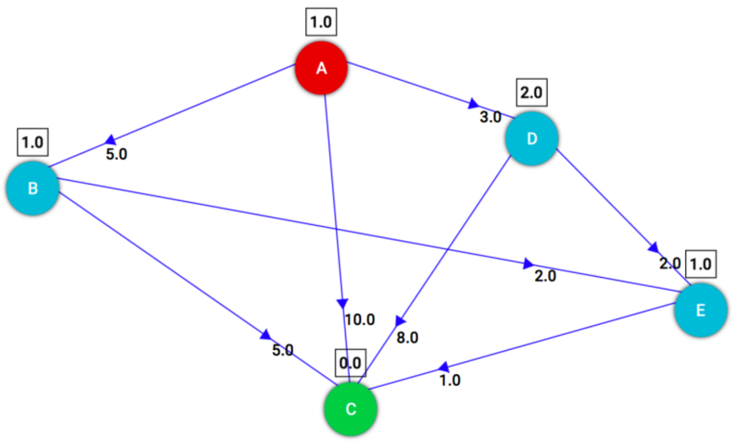
Exercise 1

Consider the following game tree where the first player is *MAX*. Show how the *min-max* algorithm works and show the *alfa-beta* cuts. Also, show which is the proposed move for the first player.



Exercise 2

Consider the following graph, where A is the starting node and C the goal node. The number on each arc is the cost of the operator for the move. Close to each node there is the heuristic evaluation of the node itself, namely its estimated distance from the goal:



a) Apply the depth-first search, and draw the developed search tree indicating for each node n the cost g(n) and the expansion order; in case of non-determinism, choose the nodes to be expanded according to the alphabetical order.

b) Apply the A* search, and draw the developed search tree indicating for each node n the function f(n) and the expansion order. In the case of non-determinism, choose the nodes to be expanded according to the alphabetical order. Consider as heuristic h(n) the one indicated in the square next to each node in the figure, that is: h (A) = 1, h (B) = 1, h (D) = 2, h (E) = 1, h (C) = 0. Is the heuristic h defined in this way admissible?

What advantage is obtained by applying A*, compared to the outcome of the depth-first search?

Exercise 3

Consider the following CSP:

X1 :: [0..2]	X1 * 10 + X2 <= 29
X2 :: [0..9]	X1 * 10 + X2> = 1
X3 :: [1..10]	X1 = (1 + X3) mod 10
X2 = (1 + X3) div 10	

Apply the Arc-consistency to the CSP, checking the constraints for each arc, up to retirement, and show the final domains of the three variables.

Exercise 4

Given the following initial state **[at(room1), have_charge, handempty]**: and actions modeled as follows:

vacuuming(room)
PRECOND: have_charge, at(room), have_vacuum_cleaner
DELETE: have_charge
ADD: vacuumed(room)

putdown_vacuum_cleaner
PRECOND: have_vacuum_cleaner
DELETE: have_vacuum_cleaner
ADD: handempty

pickup_vacuum_cleaner
PRECOND: handempty
DELETE: handempty
ADD: have_vacuum_cleaner

charge_battery
PRECOND: not have_charge
DELETE: -
ADD have_charge

go(room1, room2)
PRECOND: at(room1)
DELETE: at(room1)
ADD at(room2)

and the following goal **vacuumed(room1), vacuumed(room2)**

Solve the problem by using the POP algorithm showing threats and how to solve them.

Exercise 5

1) Model the action **vacuuming** (preconditions, effects and frame axioms), and the initial state of the exercise 4 using the Kowalsky formulation

2) Show two levels of graph plan when applied to exercise 4.

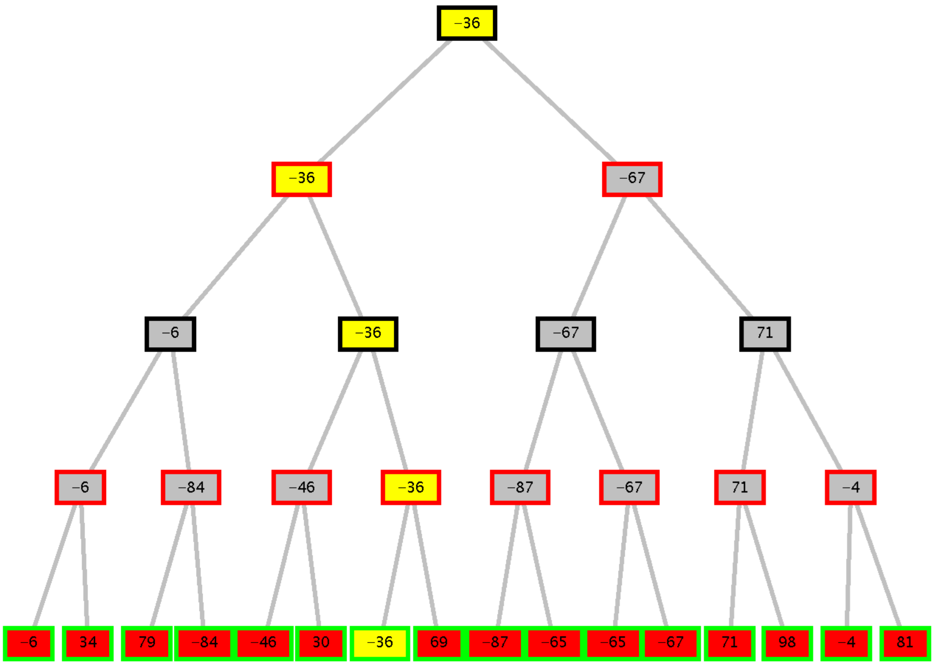
1) What are the main features of a swarm intelligence algorithm?

2) What is conditional planning and which are its main features?

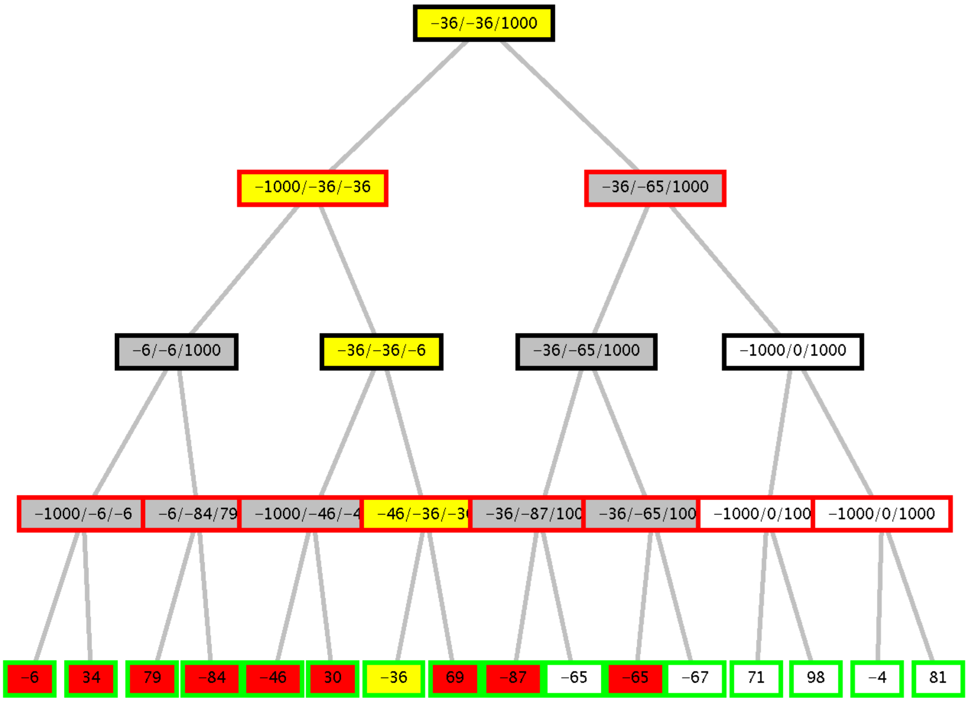
3) What is modal truth criterion and why it has been defined.

Solution

Exercise 1
Min-max

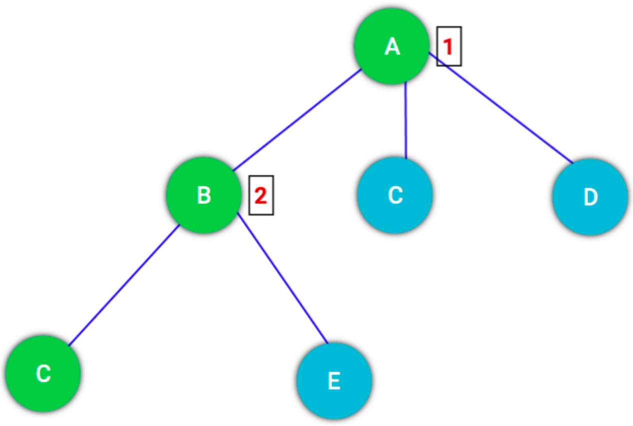


Alfa-beta :

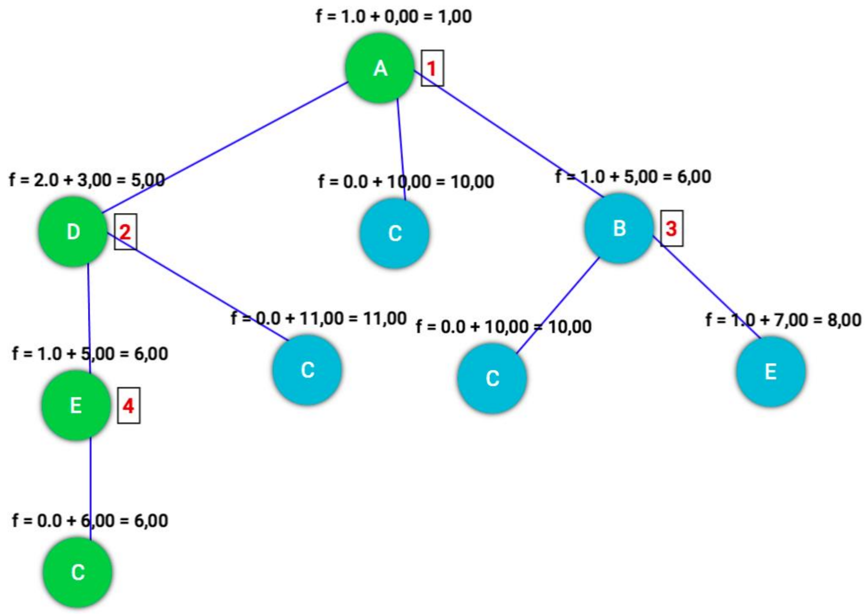


Exercise 2

Depth-first



Cost of path found (in green), ABC equal to 10.



With A*, path cost found (in green) - ADEC equal to 6 (optimal path). Admissible heuristics because it never overestimates the cost of reaching the goal, regardless of which node of the graph you consider as starting point.

Exercise 3

By applying the Arc-consistency algorithm to the obtained CSP, checking the constraints for each link in both directions, we can reduce the variables domain as following:

X1::[0..2]

X2::[0,1]

X3::[1,9,10]

Solving procedure:
Starting domains:

X1::[0..2]

X2::[0..9]

X3::[1..10]

Contraints:

a) X1*10+X2 <=29

b) X1*10+X2 >=1

c) X1= (1+X3) mod 10
d) X2= (1+X3) div 10

First iteration

Applying (a).

For each X1, there exists a X2? Yes, domains are untouched.

For each X2, there exists a X1? Yes, domains are untouched.

Applying (b)

For each X1, there exists a value in X2? Yes, domains are untouched.

For each X2, there exists a value in X1? Yes, domains are untouched.

Applying (c).

For each X1, there exists a value in X3? Si, domini untouched.

For each X3, there exists a value in X1? The only feasible values in X3, with respect to X1 domain, are: 1, 9, 10.
New domains:

X1::[0..2]

X2::[0..9]

X3::[1,9,10]

Applying (d).

For each X2, there esists value in X3? The only feasible values in X2 are: 0, 1.
New domains:

X1::[0..2]

X2::[0,1]

X3::[1,9,10]

For each X3, there esists a value in X2? Yes, domains are untouched.

Second iteration (because in the previous iteration the domains have been changed)

Applying (a):
X1*10+X2 <=29
For each X1, there esists a value in X2? Yes, domains are untouched.

For each X2 there esists a value in X1? Yes, domains are untouched.

Applying (b):
X1*10+X2 >=1
For each X1, there esists a value in X2? Yes, domains are untouched.

For each X2, there esists a value in X1? Yes, domains are untouched.

Applying (c):
X1= (1+X3) mod 10
For each X1, there esists a value in X3? Yes, domains are untouched.

For each X3 there esists a value in X1? Yes, domains are untouched.

Applying (d):
X2= (1+X3) div 10

For each X2, there esists a value in X3? Yes, domains are untouched.

For each X3, there esists a value in X1? Yes, domains are untouched.

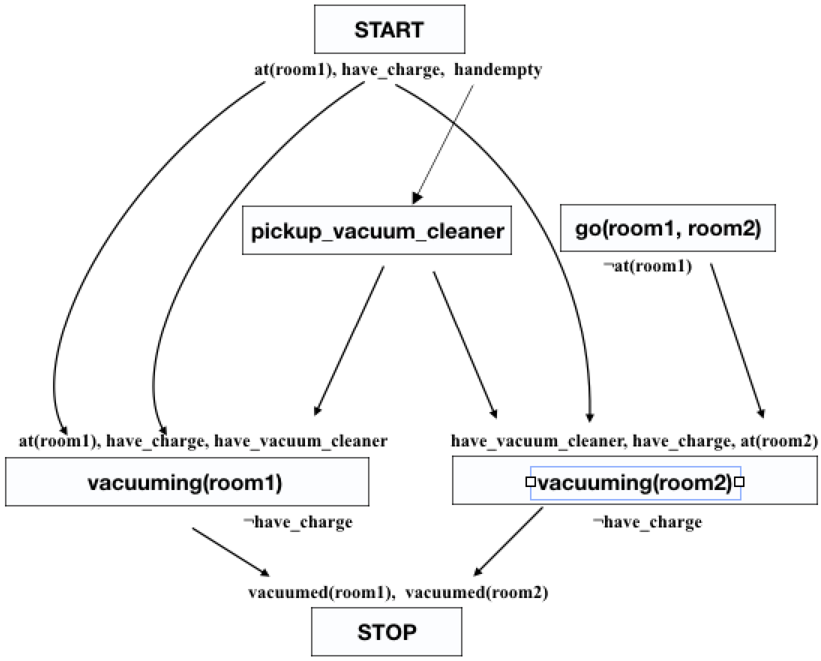
Stopping. Thus, the final domains are:

X1::[0..2]

X2::[0,1]

X3::[1,9,10]

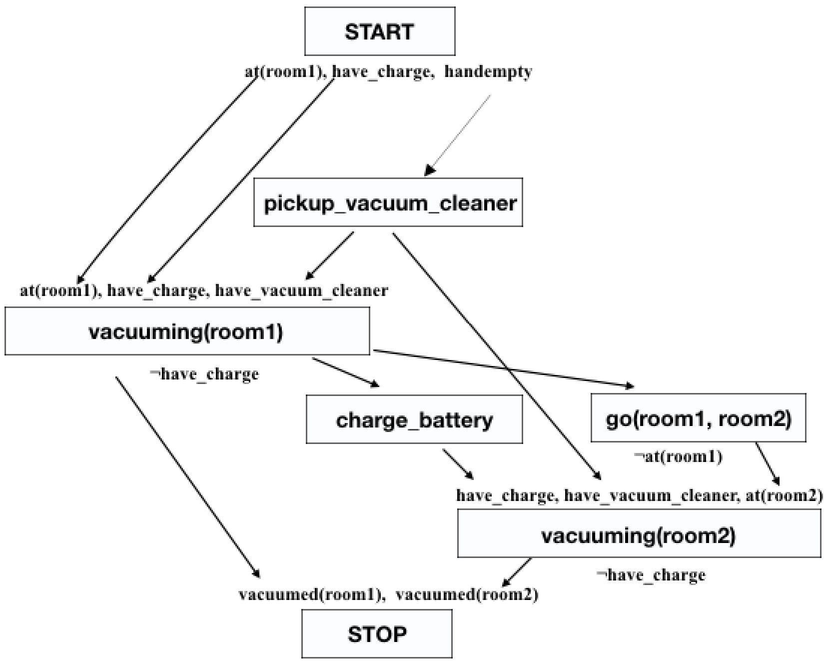
Exercise 4



The plan up to now contains threats. In particular:
<Start, vacuuming(room2), have_charge> and <Start, vacuuming(room1), have_charge> are threatened by vacuuming(room1) and vacuuming(room2) respectively. No ordering constraints can solve these threats: we need to insert a white knight charge_battery.

In addition the action go(room1, room2) threats causal link <Start, vacuuming(room1), at(room1)>

In this case demotion can solve the threat. We introduce an ordering constraint between vacuuming(room1) and go(room1, room2).



Note that we have to remove the causal link between the start and the

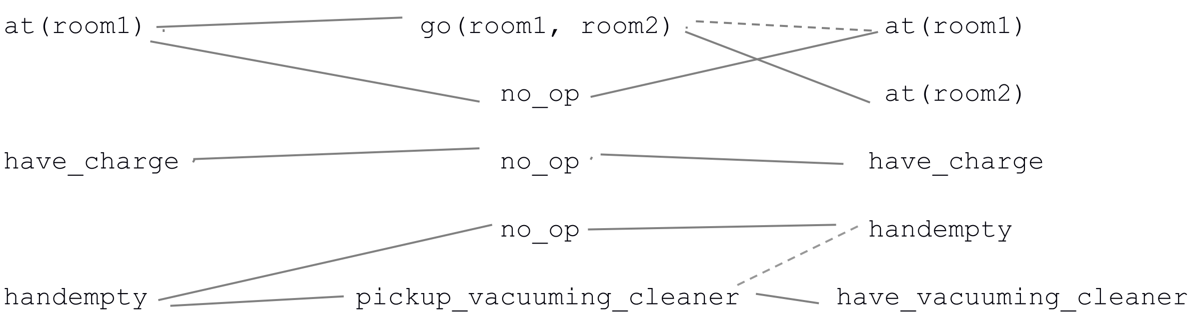
vacuuming(room2) for have charge and insert the new causal link between charge_battery and vacuuming(room2) for have_charge.

Exercise 5

1)

holds(at(room1),s0).
holds(have_charge, s0).
holds(handempty,s0).
holds(vacuumed(room), do(vacuuming(room),S))
pact(vacuuming(roomn),S):- holds(have_charge, S), holds(at(room),S), holds(have_vacuuming_cleaner,S).
holds(V,do(vacuuming(room),S*)):- holds(V,S), V\=have_charge.*

2)



The following actions are incompatible pairwise: go(room1, room2)-no_op and pickup_vacuuming_cleaner-no_op.

The following pairs of preposition are inconsistent: at(room1)-at(room2), handempty-have_vacuuming_cleaner.