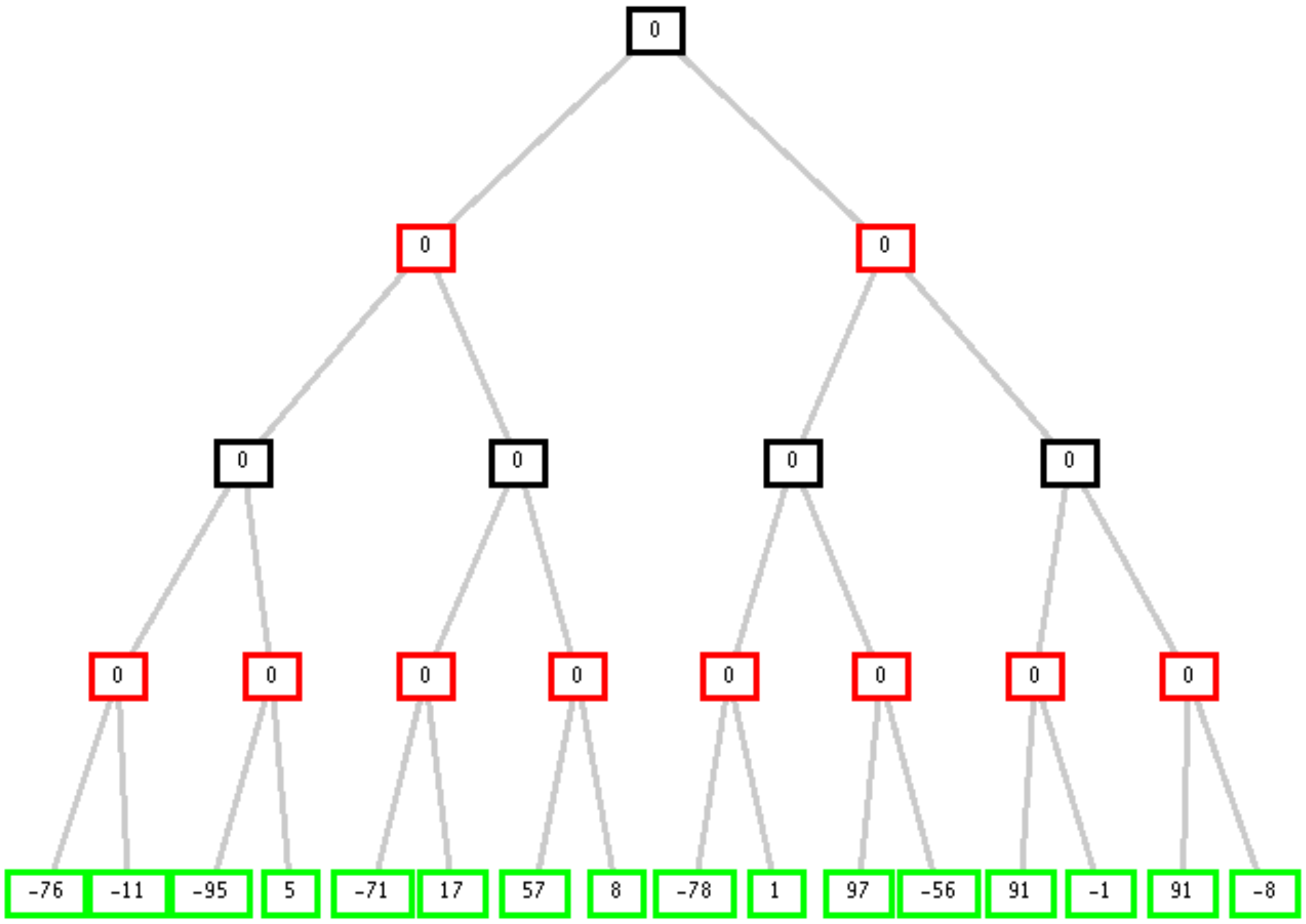
Example of Exam Foundations of AI

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

In a building there are 6 families Alberti, Bianchi, Carli, Doni, Elmi and Ferri living (not necessarily in this order) in six apartments 1,2,3,4,5,6. Apartments are on three floors, two per floor. Apartments 1 and 2 are at the first floor, 3 and 4 at the second floor and 5 and 6 at the third floor.

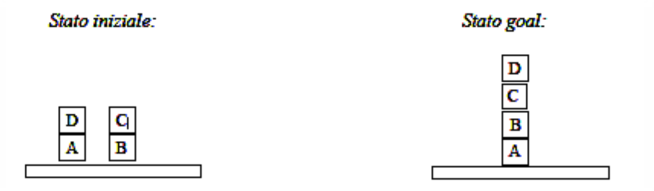
We know that Alberti and Carli do not live on the same floor, Bianchi and Ferri live at a higher floor than the one of Elmi. Doni lives at a higher floor than the one of Carli. Finally Alberti and Elmi live at the same floor.

Can we find a unique solution here? Can we find a feasible solution?

Model the problema as a CSP. Apply arc-consistency to the initial network. Then start search and apply forward checking with a first fail heuristics.

Exercise 3

Consider the following block world with initial state and goal as depicted in the figure



Available moves are the usual ones **but they can be applied to more than one block**. Don't consider loops. Show how the A* algorithm finds a solution considering as an heuristic the number of blocks that are in the "wrong position". We define *"wrong position"* when the column below a block is not the one that should be in the goal state. For example in the initial state the heuristics is 3 as all blocks except A are in the *"wrong position".*
Is the heuristic admissible?

Exercise 4

Given the following initial state **[ontable(a,p1), ontable(b,p2), ontable(d,p3), on(c,d), clear(a), clear(b), clear(c), handempty]** where a, b and c are blocks and p1, p2 and p3 positions on the table and the goal **on(c,a).** Actions are as follows

pickup(X,Pos)
PRECOND: ontable(X,Pos), clear(X), handempty
DELETE: ontable(X,Pos), clear(X), handempty
ADD: holding(X), empty(Pos)

putdown(X,Pos)
PRECOND: holding(X), empty(Pos)
DELETE: holding(X), empty(Pos)
ADD: ontable(X,Pos), clear(X), handempty

stack(X,Y)
PRECOND: holding(X), clear(Y)
DELETE: holding(X), clear(Y)
ADD: handempty, on(X,Y), clear(X)

unstack(X,Y)
PRECOND: handempty, on(X,Y), clear(X)
DELETE: handempty, on(X,Y), clear(X)
ADD: holding(X), clear(Y)

Show how the STRIPS algorithm finds a solution (only one successful path in the search tree should be shown).

Exercise 5

1. Model the action **pickup(X,Pos)** (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.

3. What are the main approaches of deductive planning. Explain the main differences.

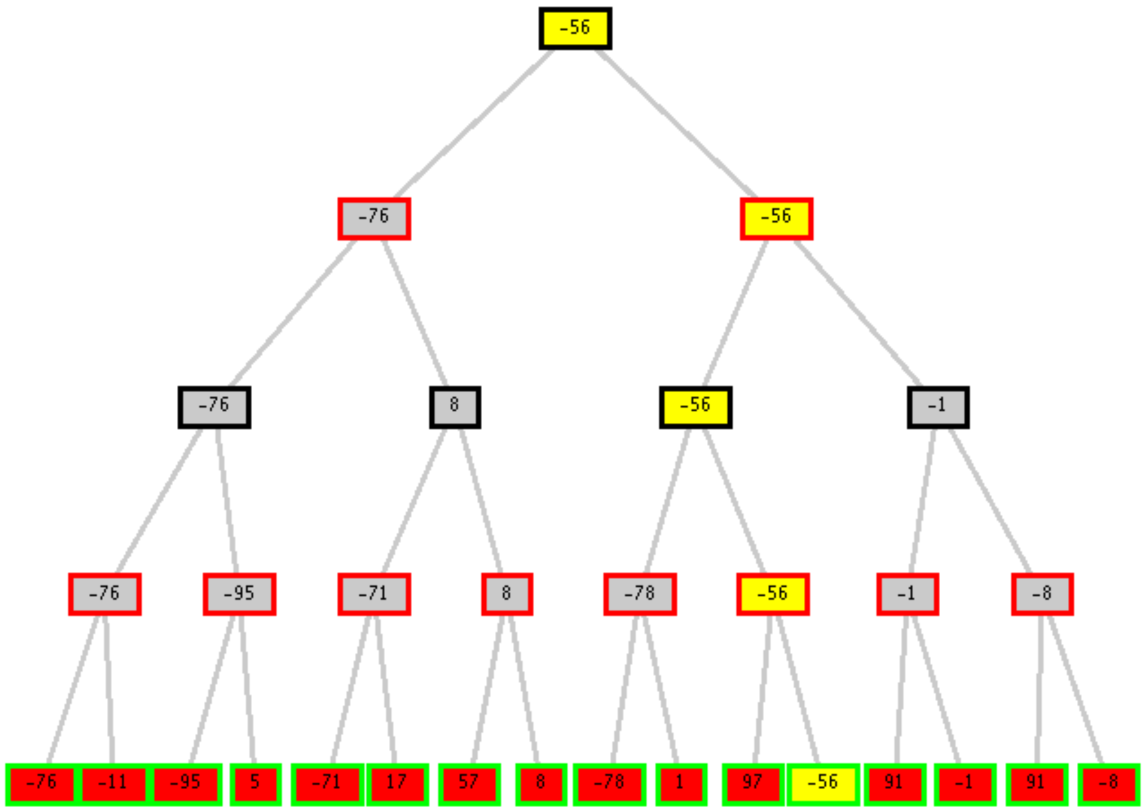
4. What are metaheuristics? Describe the main algorithms that have been presented during the course.

5. What is arc-consistency? Describe the algorithm to achieve it. Explain the properties of values that are removed from constraints and of values that are left in the domains.

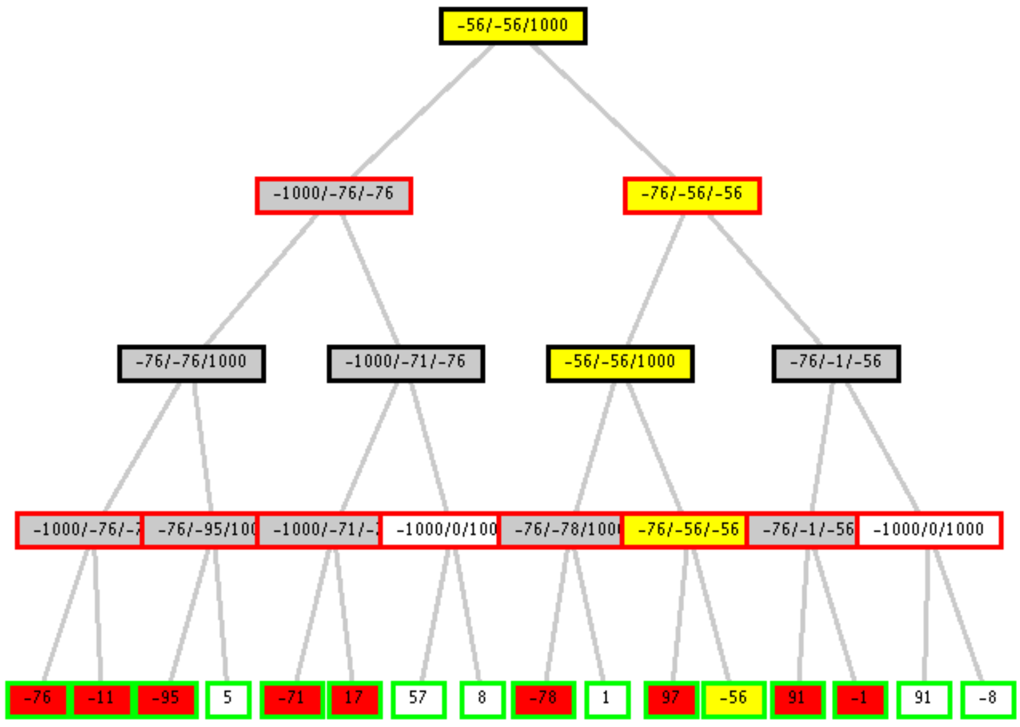
SOLUTIONS

Exercise 1

Min-Max:



Alfa-Beta:



Exercise 2

For this problem we don't have a unique solution. In fact, it is not specified if a family lives in the first or second apartment in the floor, so there are symmetric solutions.

So the problem can be modeled using in the domains the number of floors and not the number of apartments.

We have six variables A,B,C,D,E,F each with a domain containing floors [1..3]

Constraints are the following

Alberti and Carli do not live on the same floor: $A \neq C$

Bianchi lives in a higher floor than the one of Elmi: $B > E$

Ferri lives in a higher floor than the one of Elmi: $F > E$

Doni lives in a higher floor than the one of Carli: $D > C$

Alberti and Elmi live on the same floor: $A = E$

Arc-consistency on the initial network

A::[1,2], B::[2,3], C::[1,2], D::[2,3], E::[1,2] F::[2,3]

All variables have two values in their domains, so we select one randomly

A = 1

Forward checking

B::[2,3], C::[2] , D::[2,3], E:: [1], F::[2,3]

C = 2

B::[2,3], D::[3], E:: [1], F::[2,3]

D = 3 no changes

B::[2,3], E:: [1], F::[2,3]

E = 1 no changes

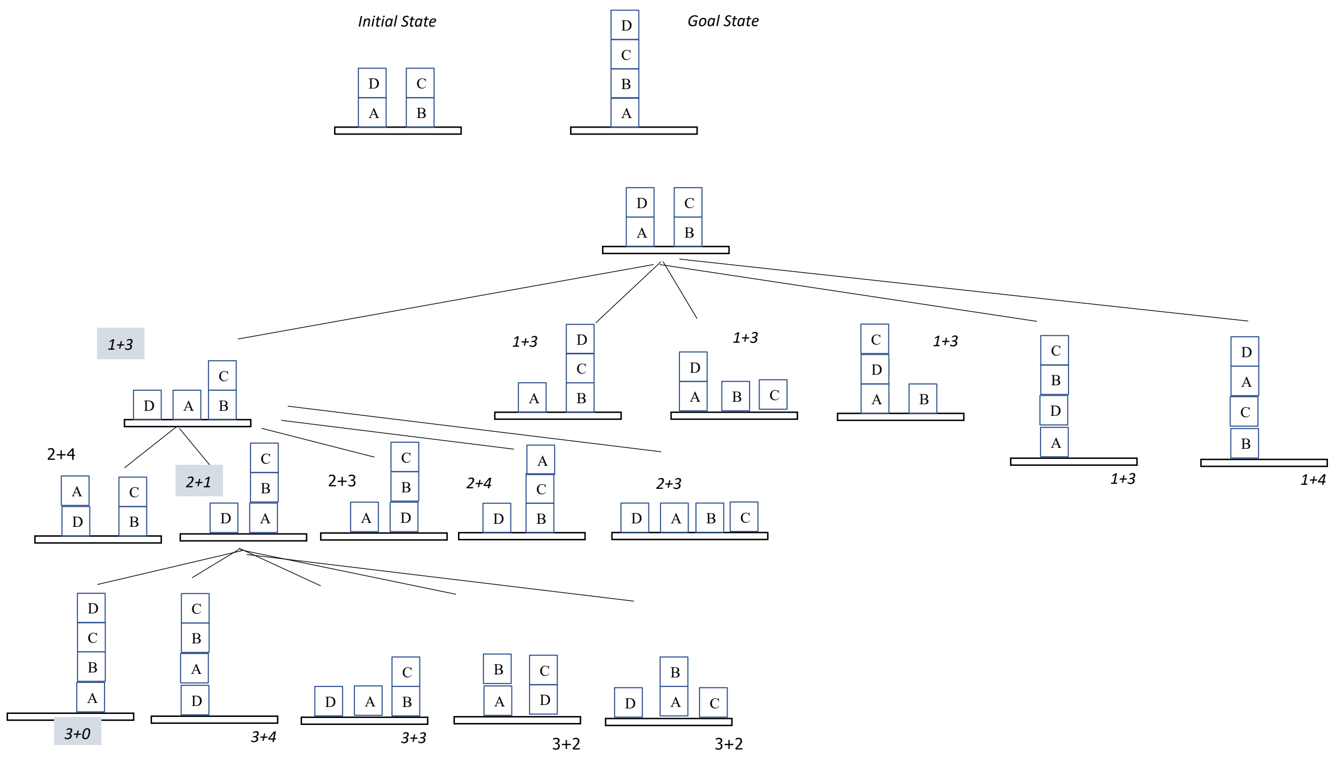
B::[2,3], F::[2,3]

B = 2

F::[3]

F = 3

Exercise 3



The algorithm finds the solution at depth 3, which is not optimal. This is because the heuristics is not admissible. In fact, there is a goal at depth 2.

Exercise 4

///////////////////

///

state	goal
ontable(a,p1)	handempty
ontable(b,p2)	on(c,Y)
ontable(d,p3)	clear(c)
on(c,d)	handempty ∧ on(c,Y) ∧ clear(c)
clear(a)	unstack(c,Y)
clear(b)	holding(c) ∧ clear(a)
~~clear(c)~~	

state	goal
ontable(a,p1)	on(c,Y)
ontable(b,p2)	clear(c)
ontable(d,p3)	handempty ∧ on(c,Y) ∧ clear(c)
on(c,d)	unstack(c,Y)
clear(a)	holding(c) ∧ clear(a)
clear(b)	stack(c,a)
~~clear(c)~~	

Y/d

state	goal
ontable(a,p1)	clear(c)
ontable(b,p2)	handempty ∧ on(c,d) ∧ clear(c)
ontable(d,p3)	unstack(c,d)
on(c,d)	holding(c) ∧ clear(a)
clear(a)	stack(c,a)
clear(b)	
~~clear(c)~~	

state	goal
ontable(a,p1)	handempty ∧ on(c,d) ∧ clear(c)
ontable(b,p2)	unstack(c,d)
ontable(d,p3)	holding(c) ∧ clear(a)
on(c,d)	stack(c,a)
clear(a)	
clear(b)	
~~clear(c)~~	

state

```
ontable(a,p1)
ontable(b,p2)
ontable(d,p3)
on(c,d)
clear(a)
clear(b)
clear(c)
```

goal

```
unstack(c,d)
holding(c) ∧ clear(a)
stack(c,a)
```

---

state

```
ontable(a,p1)
ontable(b,p2)
ontable(d,p3)
clear(a)
clear(b)
clear(d)
holding(c)
```

goal

```
holding(c) ∧ clear(a)
stack(c,a)
```

---

state

```
ontable(a,p1)
ontable(b,p2)
ontable(d,p3)
clear(a)
clear(b)
clear(d)
holding(c)
```

goal

```
stack(c,a)
```

---

state

```
ontable(a,p1)
ontable(b,p2)
ontable(d,p3)
on(c,a)
clear(b)
```

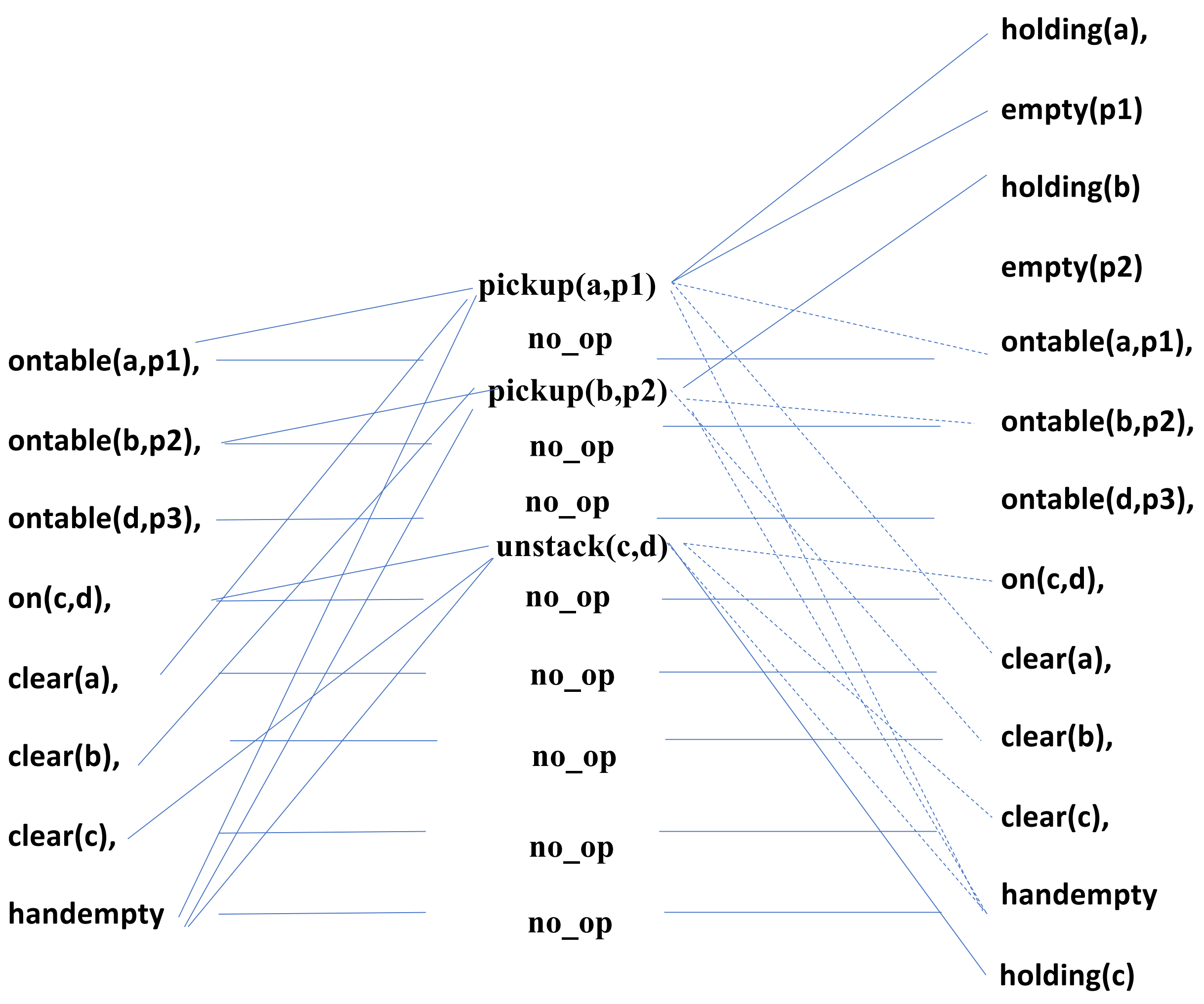
goal

Exercise 5

1) holds(ontable(a,p1), s0).
   holds(ontable(b,p2), s0)…..same for the other properties of the initial state

   holds(holding(X), do(pickup(X,P),S)).
   pact(pickup(X,P),S):- holds(clear(X), S), holds(ontable(X,P),S), holds(handempty,S).
   *holds(V,do(*pickup(X,P),S*):- holds(V,S), V\=clear(X), V\=handempty, V\=ontable(X,P).*

2)



Unstack and the two pickup are incompatible as all of them deletes handempty which is a precondition of all of them. There is incompatibility with the corresponding no-op for all the delete lists.

Clearly the three holding are mutually incompatible. Also holding(a) is incompatible with ontable(a) and clear(a) and the same holds for holding(b) and holding(c) which is incompatible with on(c,d) and with clear(c).