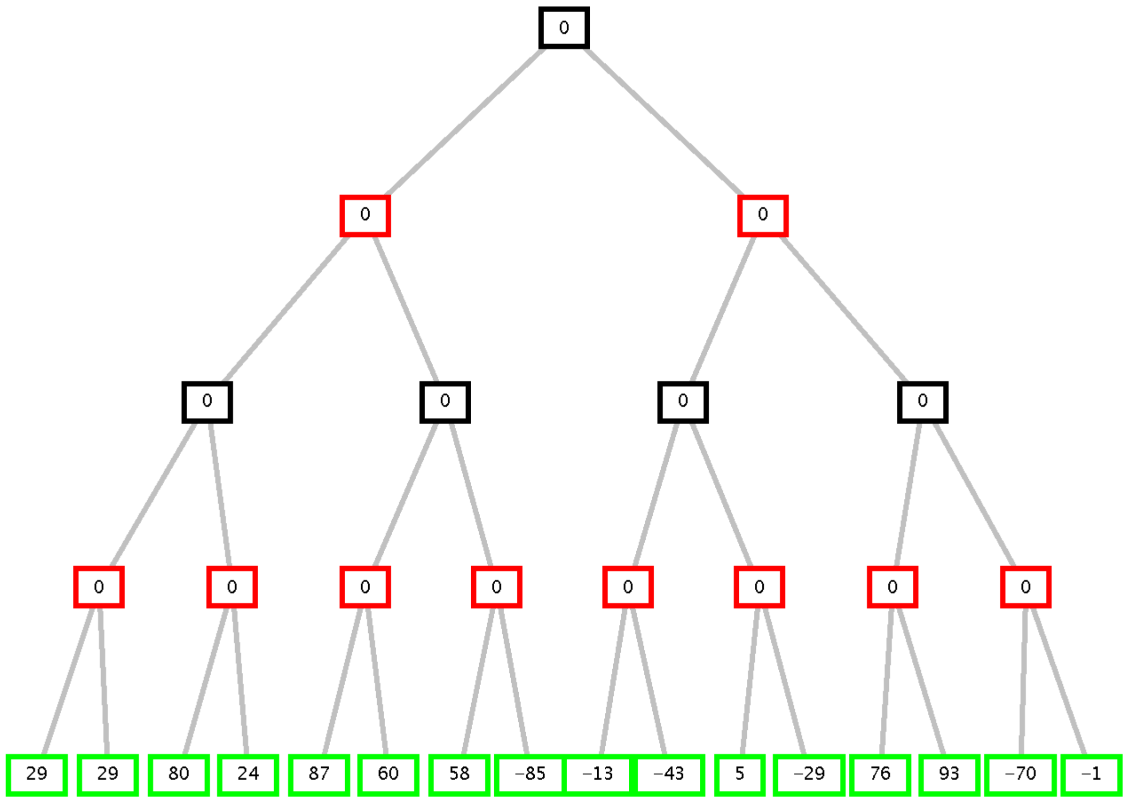
EXAMPLE EXAM OF FOUNDAMENTALS OF AI – FIRST MODULE

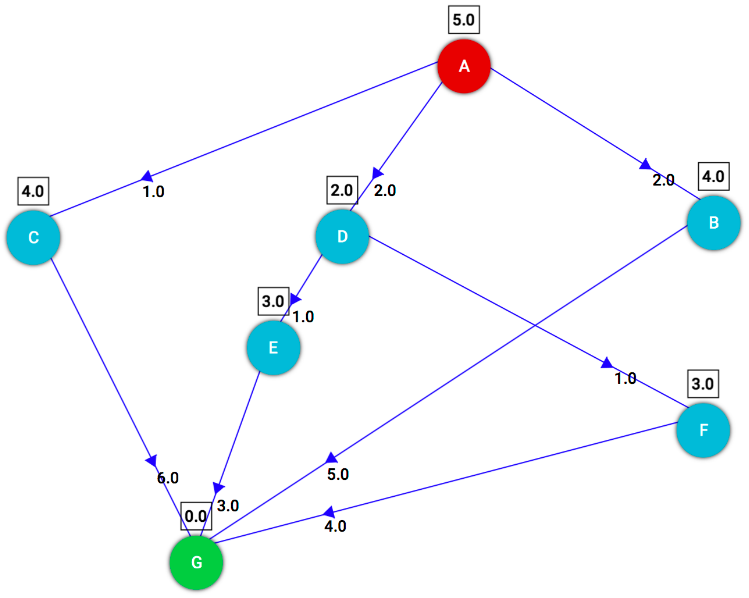
Exercise 1

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 G 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 to the goal:



a) Show the search tree generated by the A* algorithm along with the order of espansion of nodes. In case of ties, chose the node to expand in alfabetical order.
b) Is the heuristic admissible?
c) Which is the cost of the path found by A* and the number of nodes expanded?

Esercizio 3

Given the following CSP:

A::[1, 2, 3, 4, 5, 6]
B::[1, 2, 3, 4, 5, 6]
C::[1, 2, 3, 4, 5, 6]
A>=B+1
B>=C-3

Find the first solution through tree search, by applying forward checking, using alphabetical order of variables and lexicografic order of values.

Exercise 4

Given the following initial state **[at(locationA), have_battery, handempty]**: and actions modelled as follows:

take_picture(Location)
PRECOND: have_battery, at(Location), have_camera
DELETE: have_battery
ADD: picture(Location)

putdown_camera
PRECOND: have_camera
DELETE: have_camera
ADD: handempty

pickup_camera
PRECOND: handempty
DELETE: handempty
ADD: have_camera

charge_battery
PRECOND: not have_battery
DELETE: -
ADD have_battery

go(Location1, Location2)
PRECOND: at(Location1)
DELETE: at(Location1)
ADD at(Location2)

and the following goal **picture(locationA), picture(locationB)**
Solve the problem by using the POP algorithm showing threats and how to solve them.

Exercise 5

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

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

3) What is conditional planning and what are its main limitations?

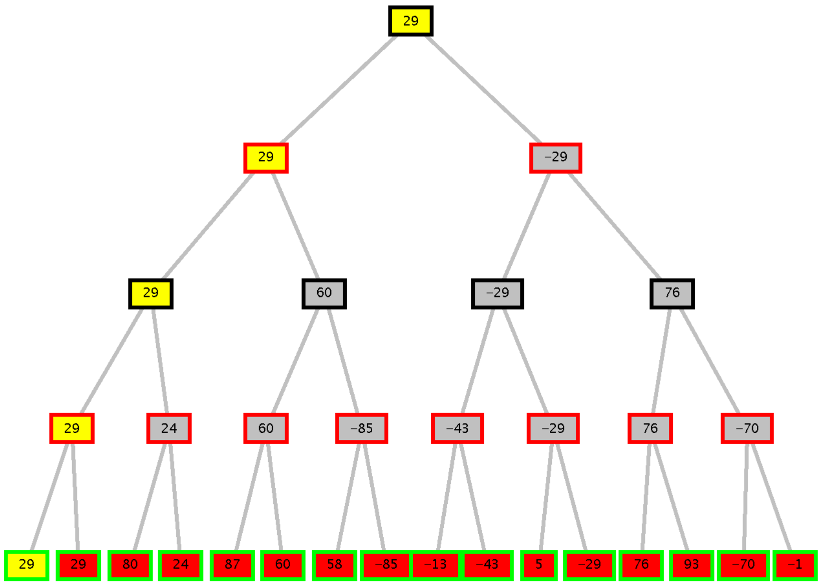
4) What is ant colony optimization?

5) What are the main features of iterative deapening?

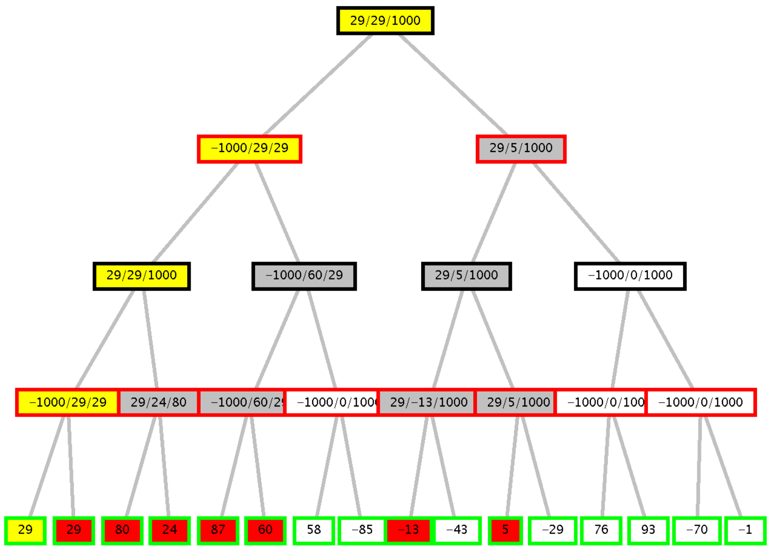
Solution

Exercise 1

Min-max

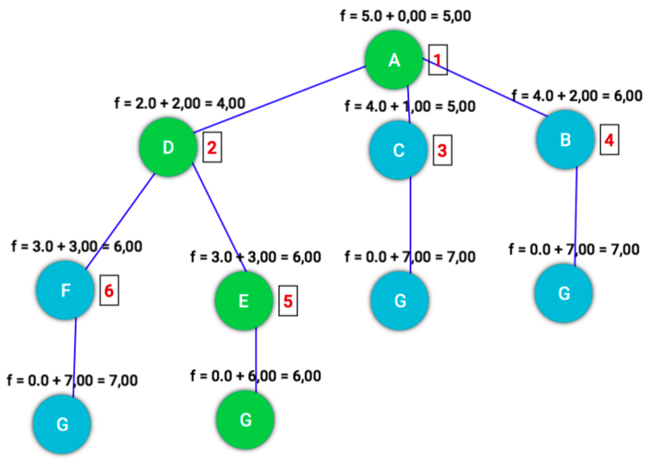


Alfa-beta cuts:



Exercise 2

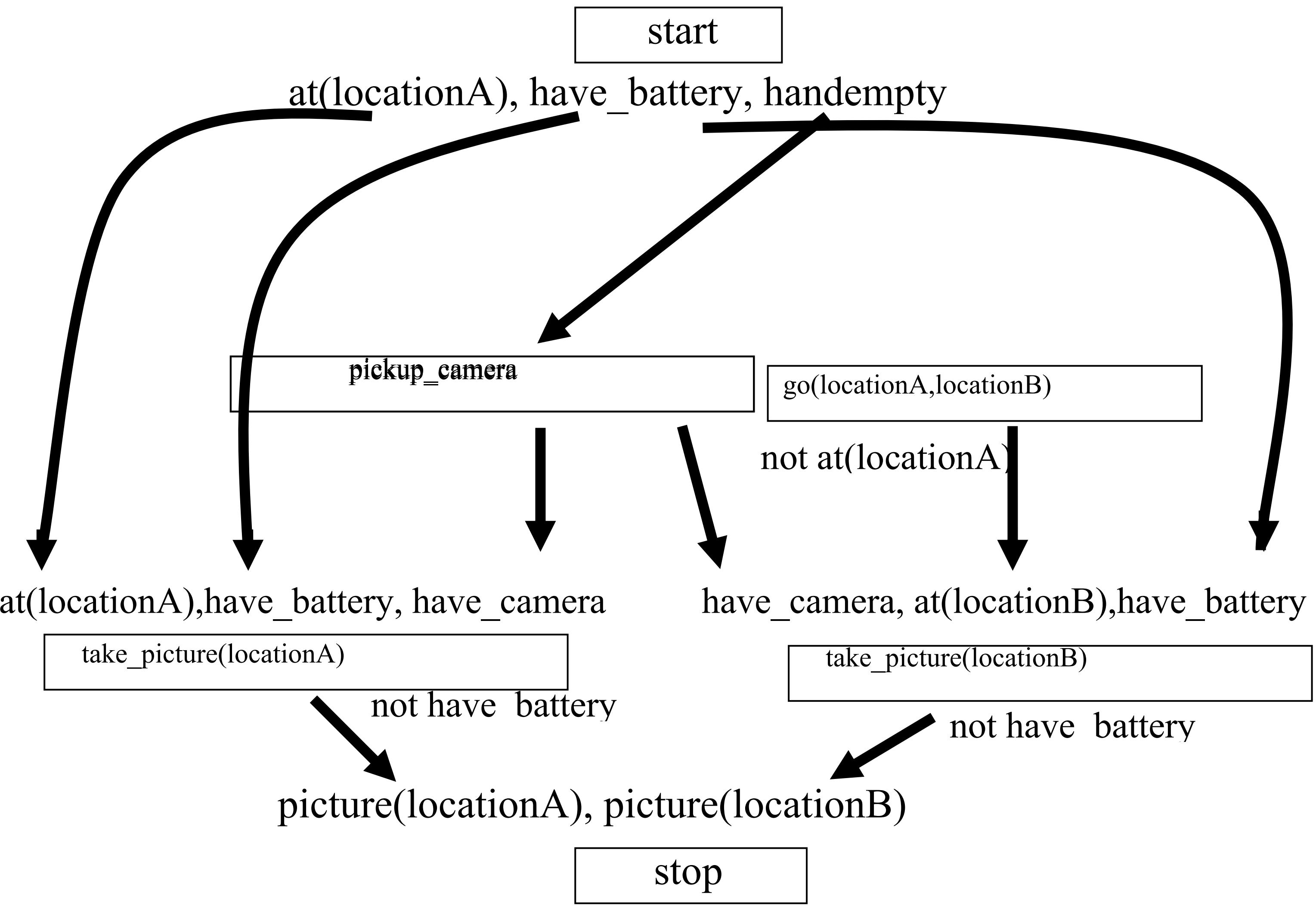
A* (Admissible heuristics; cost of the path: 6; number of expanded node: 6, with a square box close to them with the order of expansion)



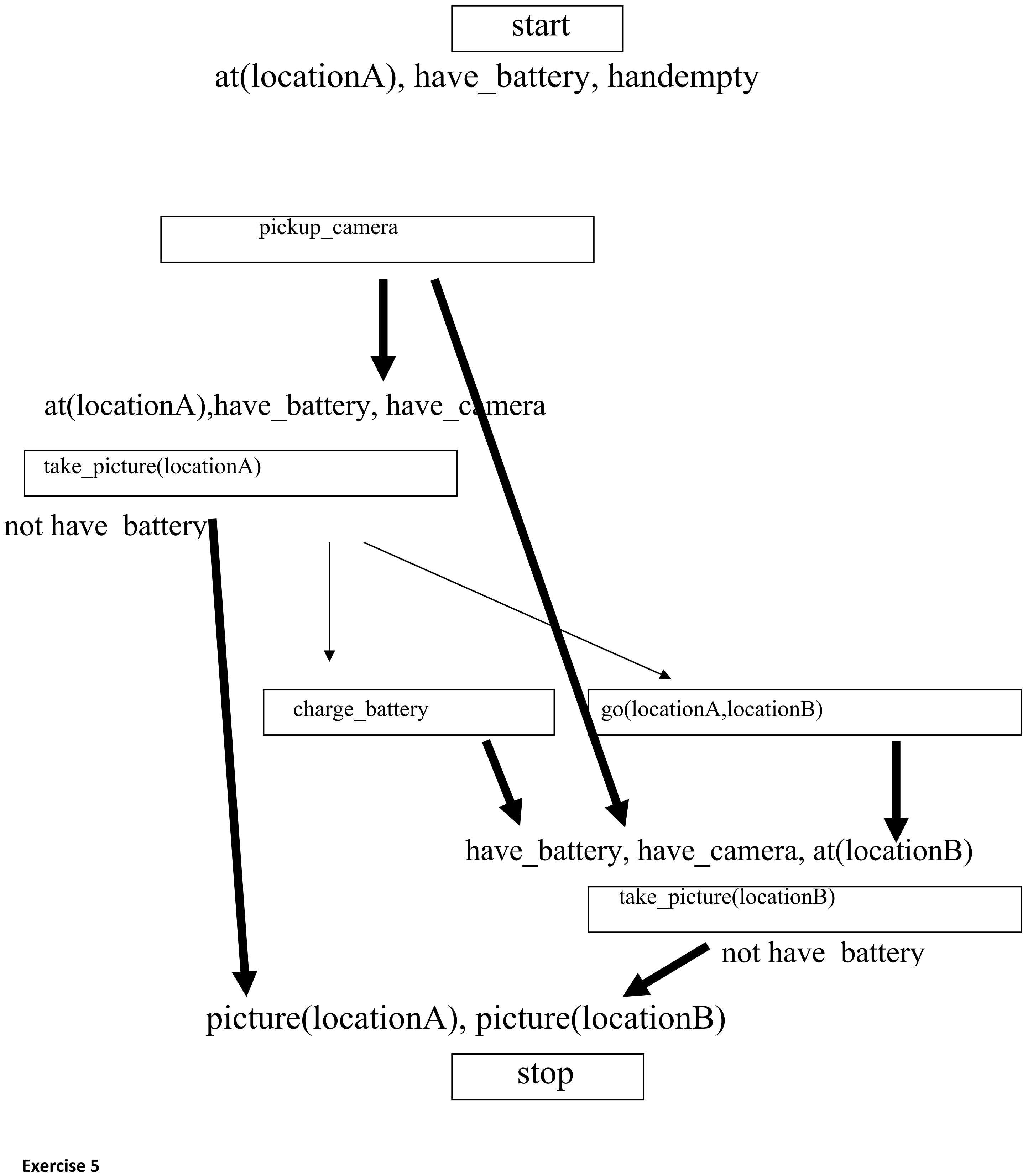
Exercise 3

	A	B	C
Backtracking	A=1	Fail	[1...6]
Labeling e FC	A=2	[1]	[1...6]
Labeling e FC	A=2	B=1	[1...4]
Labeling e FC	A=2	B=1	C=1

Exercise 4


start
at(locationA), have_battery, handempty
pickup_camera
go(locationA,locationB)
not at(locationA)
at(locationA),have_battery, have_camera
have_camera, at(locationB),have_battery
take_picture(locationA)
take_picture(locationB)
not have battery
not have battery
picture(locationA), picture(locationB)
stop

The plan up to now contains threats. In particular:
<Start, take_picture(locationB), have_battery> and <Start, take_picture(locationA), have_battery>
are threatened by take_picture(locationA) and take_picture(locationB) respectively.
No ordering constraints can solve these threats: we need to insert a white knight charge_battery.
In addition the action go(locationA, locationB) threats causal link
<Start, take_picture(locationA), at (locationA)>
In this case demotion can solve the threat. We introduce an ordering constraint between
take_picture(locationA) and go(locationA, locationB).

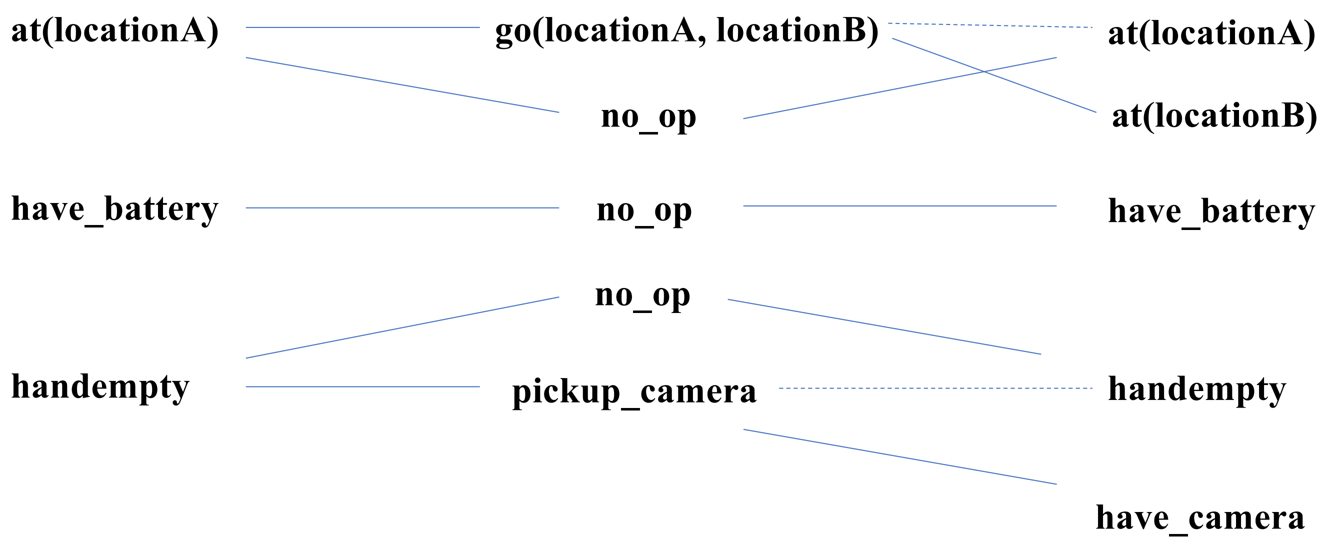

start
at(locationA), have_battery, handempty
pickup_camera
at(locationA),have_battery, have_camera
take_picture(locationA)
not have battery
charge_battery
go(locationA,locationB)
have_battery, have_camera, at(locationB)
take_picture(locationB)
not have battery
picture(locationA), picture(locationB)
stop

Exercise 5

1)

```
holds(at(location1),s0).
holds(have_battery, s0).
holds( handempty,s0).
holds(picture(Location), do(take_picture(Location),S))
pact(take_picture(Location),S):- holds(have_battery, S), holds(at(Location),S), holds(have_camera,S).
holds(V,do(take_picture(Location),S)):-  holds(V,S), V\=have_battery.
```

2)



go(locationA,locationB) incpmpatible with no_op on at(locationA)
handempty and have_camera are incompatible
at(locationA) and at(locationB) are incompatible