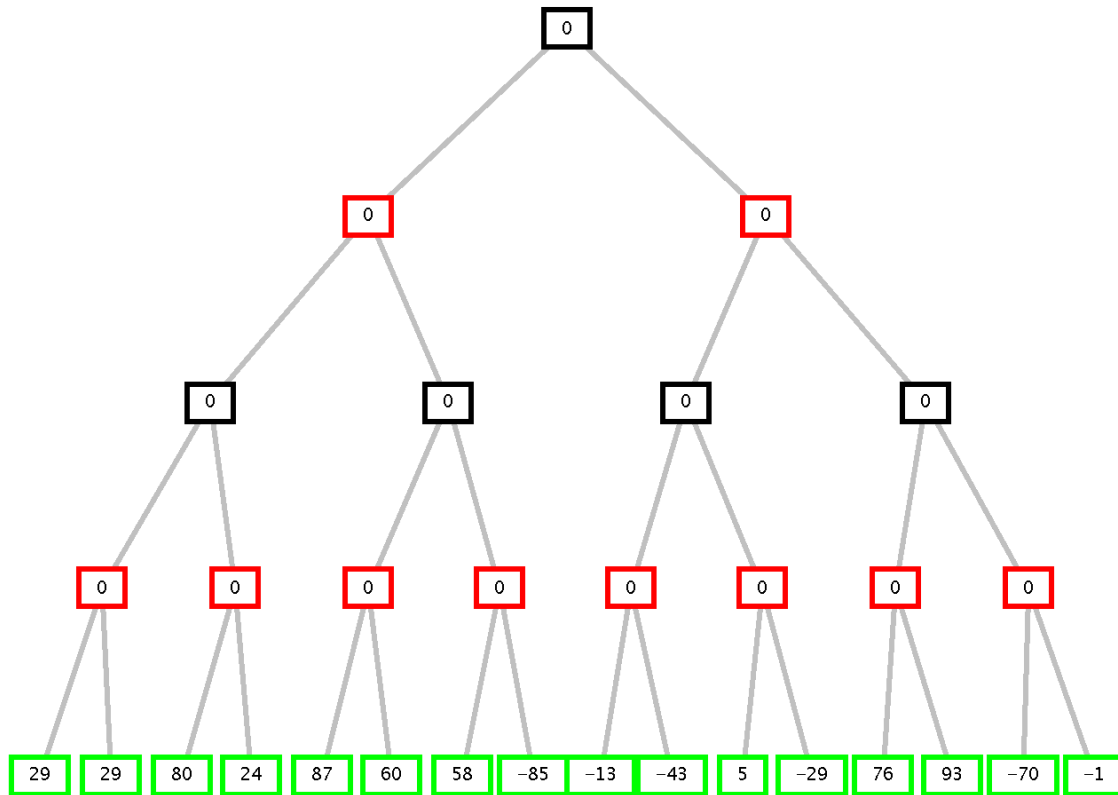


## EXAMPLE EXAM OF FUNDAMENTALS OF AI – FIRST MODULE

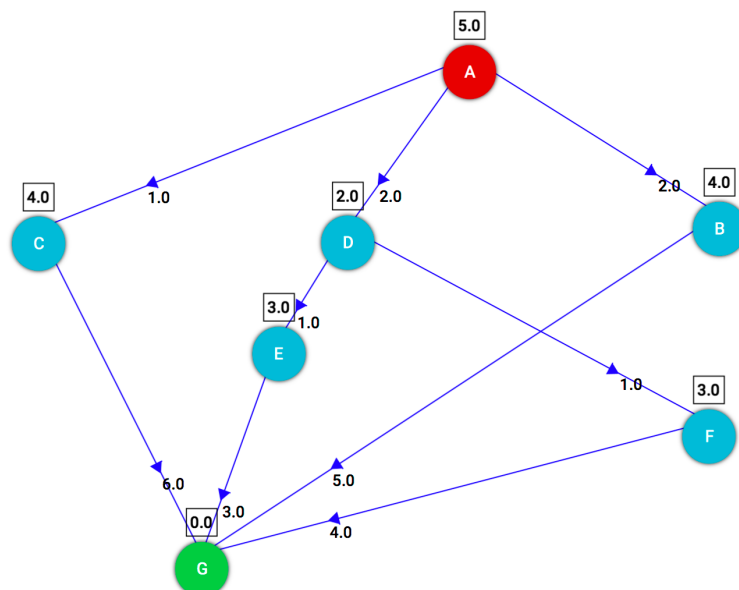
### Exercise 1

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 from the goal:



- Show the search tree generated by the A\* algorithm along with the order of expansion of nodes. In case of ties, chose the node to expand in alphabetical order.
- Is the heuristic admissible?
- Which is the cost of the path found by A\* and the number of nodes expanded?

### Exercise 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]

$A \geq B + 1$

$B \geq C - 3$

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

### Exercise 4

Given the following initial state **[at(locationA), have\_battery, handempty]**: and actions modeled 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.

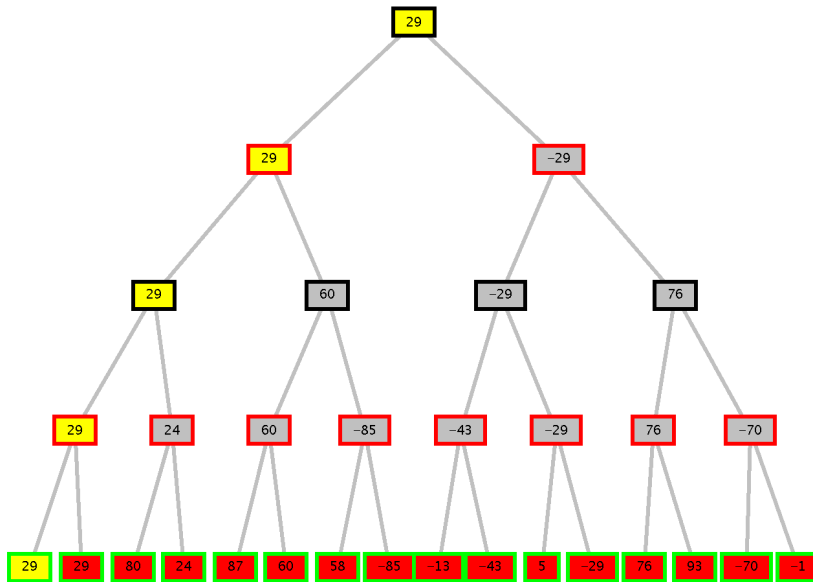
### Exercise 5

- 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 deepening?

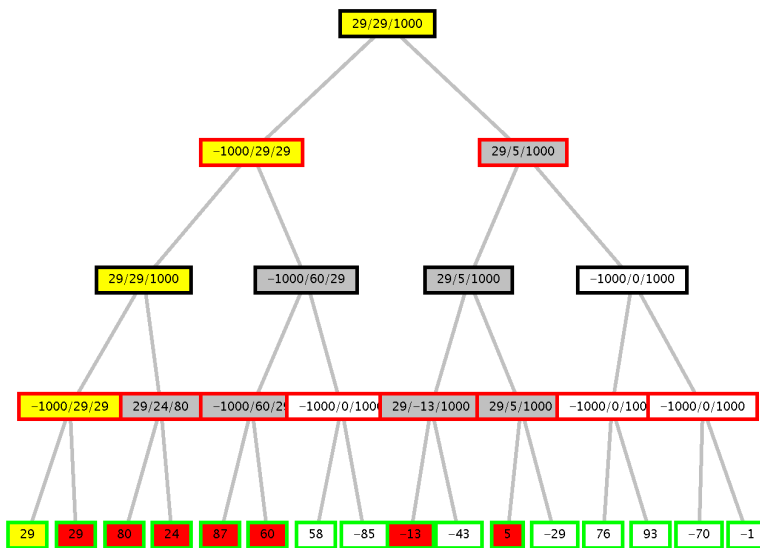
## Solution

### Exercise 1

Min-max

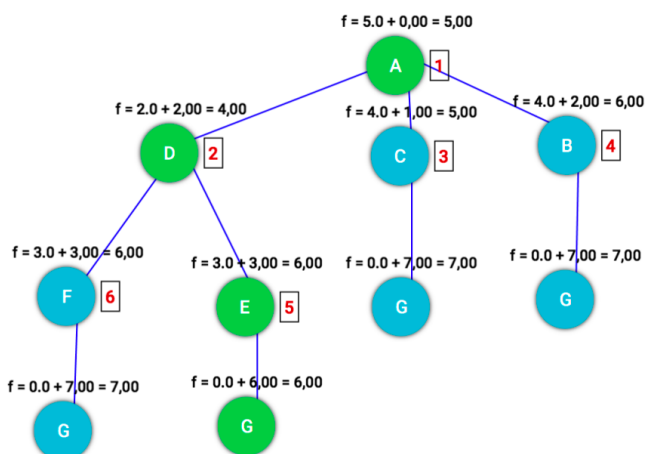


Alfa-beta cuts:



### 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)

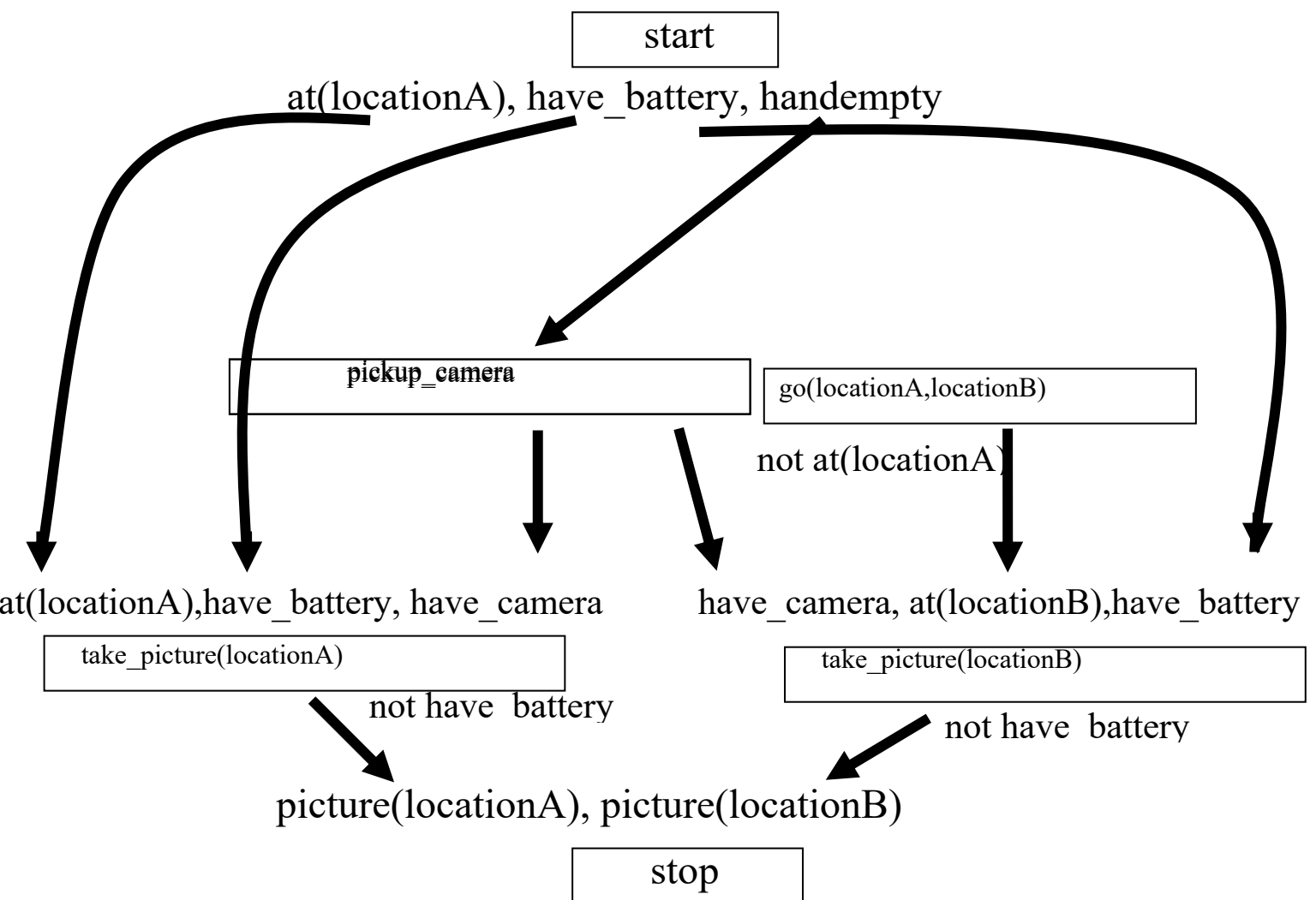


### Exercise 3

Backtracking  
Labeling e FC  
Labeling e FC  
Labeling e FC

A	B	C
A=1	Fail	[1...6]
A=2	[1]	[1...6]
A=2	B=1	[1...4]
A=2	B=1	C=1

### Exercise 4



The plan up to now contains threats. In particular:

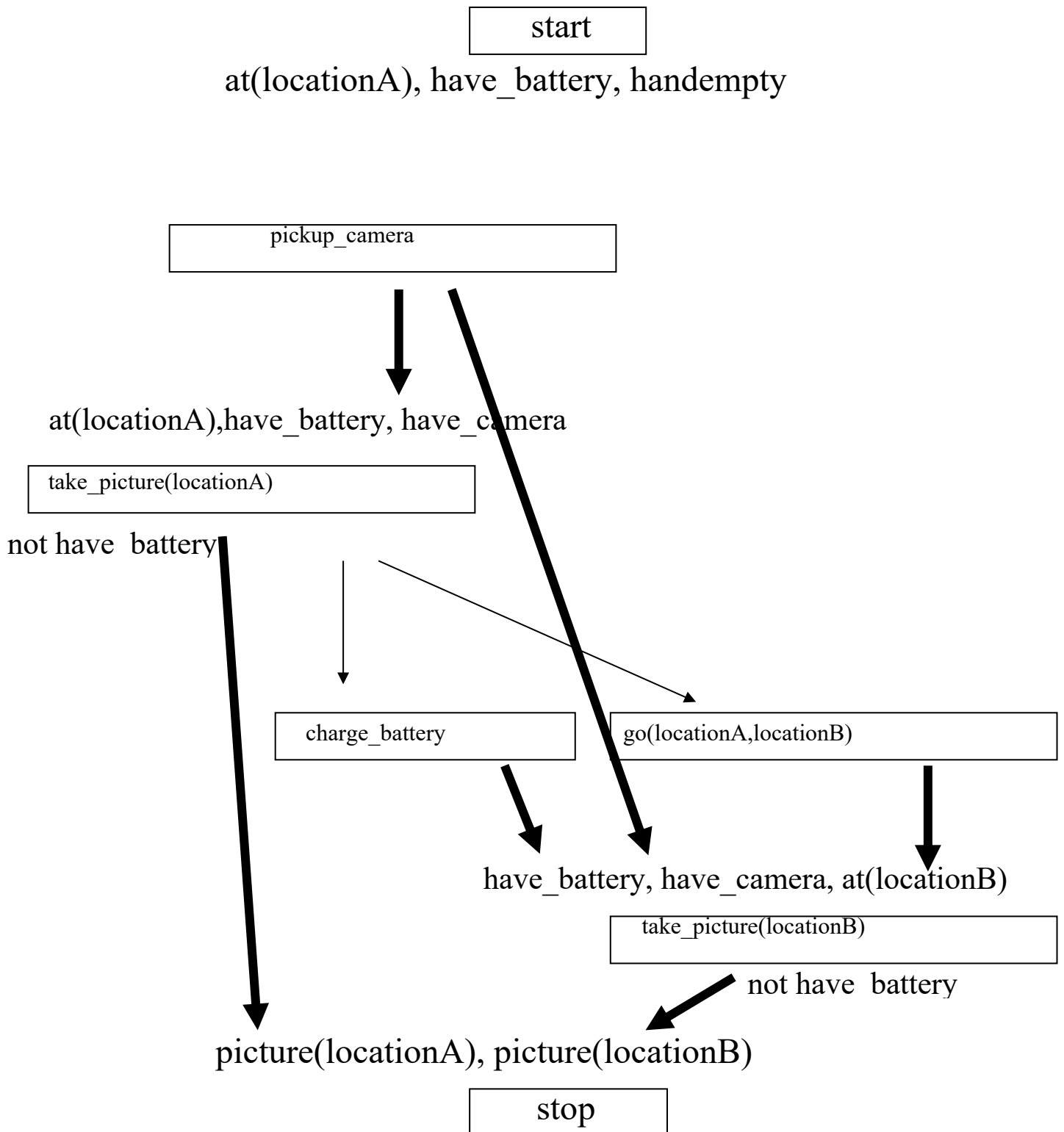
<Start, take\_picture(locationB), have\_battery> and <Start, take\_picture(locationA), have\_battery> are threatened by take\_picture(locationA) and take\_picture(locationB) respectively.

No ordering constraints can solve these threats: we need to insert a white knight charge\_battery.

In addition the action go(locationA, locationB) threatens 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).



## Exercise 5

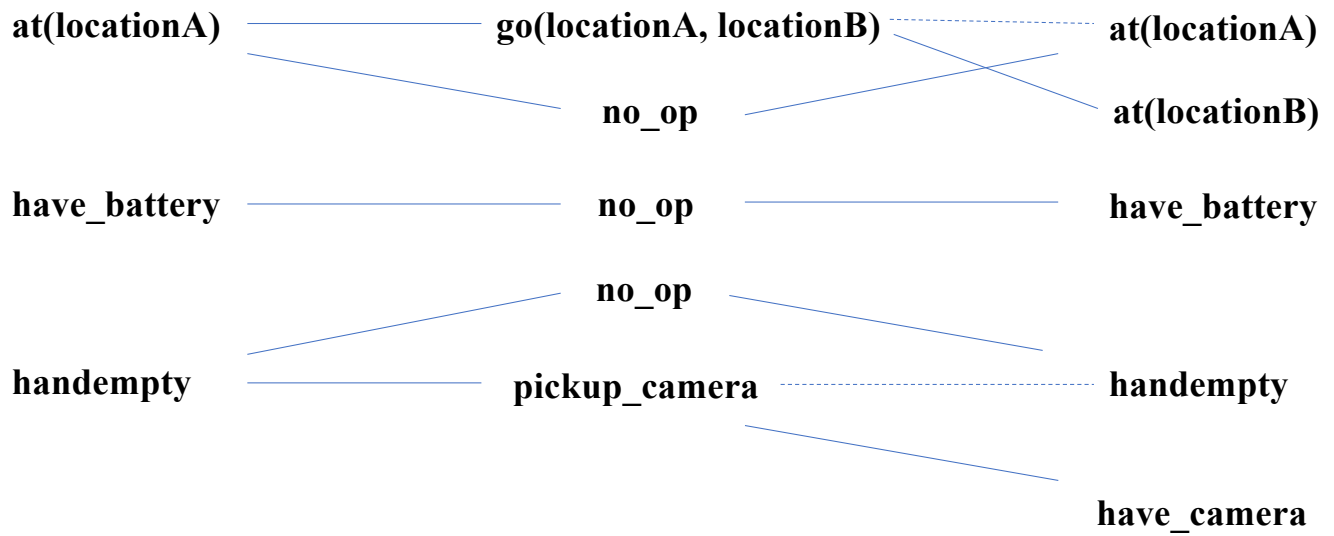
1)

```

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) incpmptible with no\_op on at(locationA)**

**handempty and have\_camera are incompatible**

**at(locationA) and at(locationB) are incompatible**