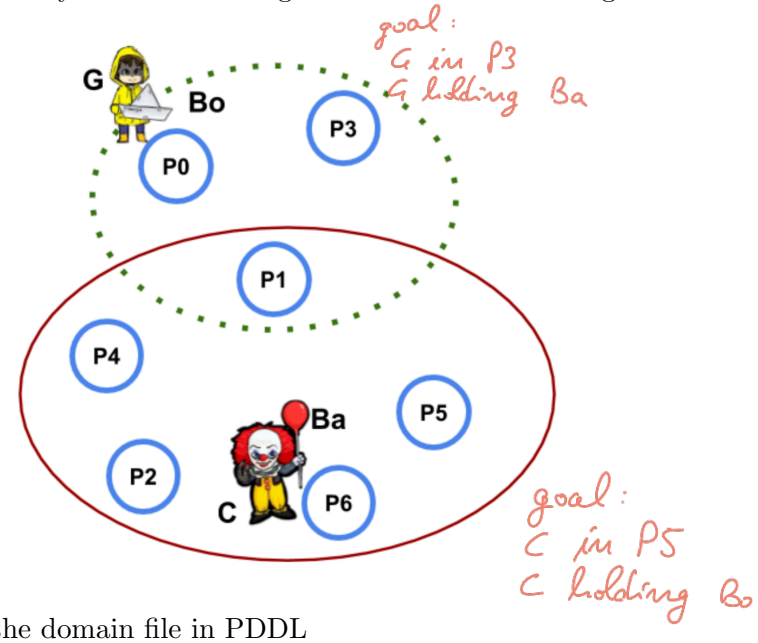
Exercise 1 (8 points)

Group A. A clown C and Georgie G are tired of playing with their toys and they want to exchange them. G wants to play with the balloon Ba while, C wants to play with the boat Bo. Unfortunately, they are both very shy and they never want to be at the same place Pi. Hence, they have to find a common place (for example $P1$) where to drop and collect objects (they can drop an object in a place and move into another place). G and C cannot hold two objects at the same time. The environment is depicted in the figure, G can only move in places within the dotted line; while C can only move within the continuous line. Places within the same set are all connected (i.e. $\{P0, P1, P3\}$ for the dotted set and $\{P1, P2, P4, P5, P6\}$ for the other). The figure shows the initial state where G holds Bo and he is at $P0$, and C holds Ba and he is at $P6$. The goal state is represented by G at $P3$ holding Ba and C at $P5$ holding Bo.



(a) Define the problem and the domain file in PDDL

(b) Show one possible sequence of actions to a goal state including all the states in the sequence

(c) Draw the first 3 steps of the tree generated by forward search assuming a perfect heuristic (a heuristic choosing the move in the plan given above). Show all the actions applicable at each of the traversed states, and the state reached.

(a)

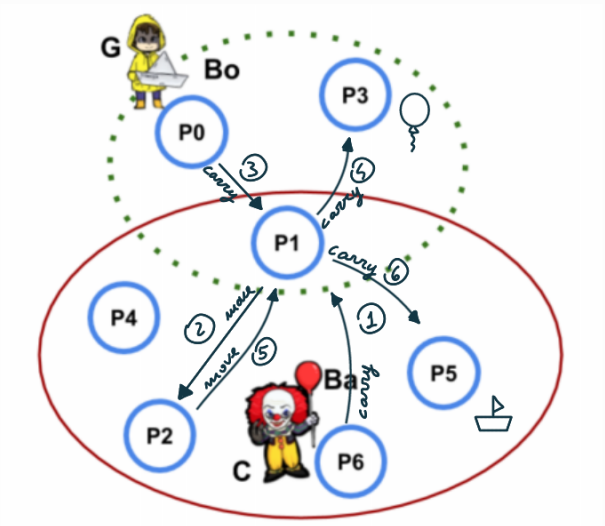
```
(define (domain IT_domain)
    (:requirements :STRIPS)
    (:predicates (at ?what ?where) ; is the entity (agent/object) in the target pos?
                 (empty ?pos)      ; is there already an agent in ?pos ?
                 (valid ?agent ?pos) ; is the position pos? valid for ?agent ?
                 (is_object ?what)
    )

    (:action move
        :parameters (?agent ?from ?to)
        :precondition (and (at ?agent ?from)(empty ?to)(valid ?agent ?to))
        :effect (and (at ?agent ?to)(not(at ?agent ?from))
                (empty ?from)(not(empty ?to))
    )
    (:action carry
        :parameters (?agent ?what ?from ?to)
        :precondition (and (at ?agent ?from)(at ?what ?from)
                      (valid ?agent ?to)(empty ?to))
        :effects (and (at ?agent ?to)(at ?what ?to)
                 (not (at ?what ?from))(not (at ?agent ?from))
                 (empty ?from)(not (empty ?to)))
    )
)
```

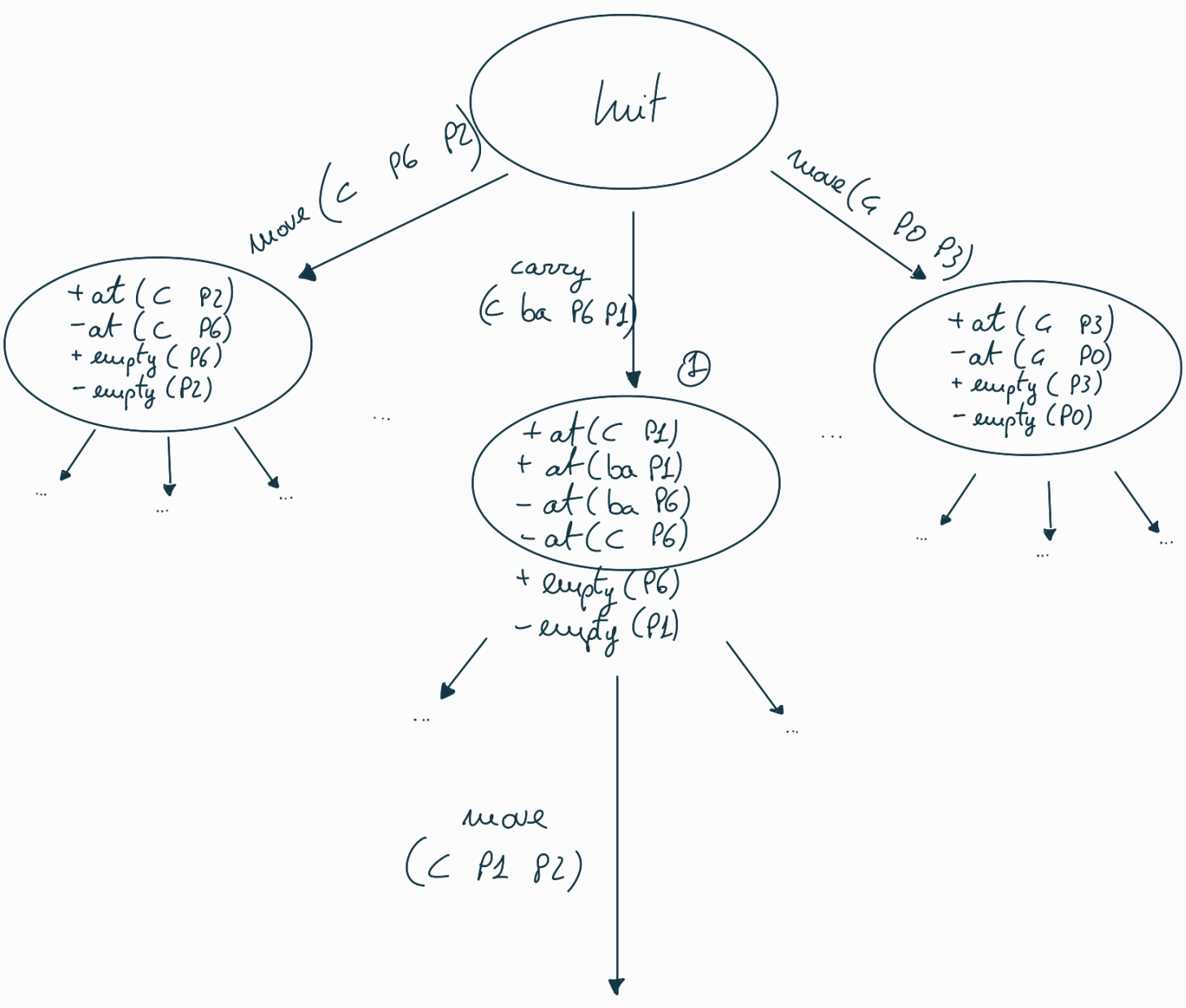
```
(define (problem IT_problem)
        (:domain IT_domain)
        (:objects G C Ba Bo P0 P3 P1 P2 P4 P5 P6 )
        (:init (at G P0) (at C P6)
               (at Bo P0) (at Ba P6)
               (empty P1)(empty P3) (empty P4)(empty P2)(empty P5)
               (valid G P0)(valid G P3)(valid G P1)
               (valid C P1)(valid C P2)(valid C P4)(valid C P5)(valid C P6)
               (is_object Ba)(is_object Bo)
        )
        (:goal (at G P3) (at C P5)
               (at Bo P5) (at Ba P3)
        )
```

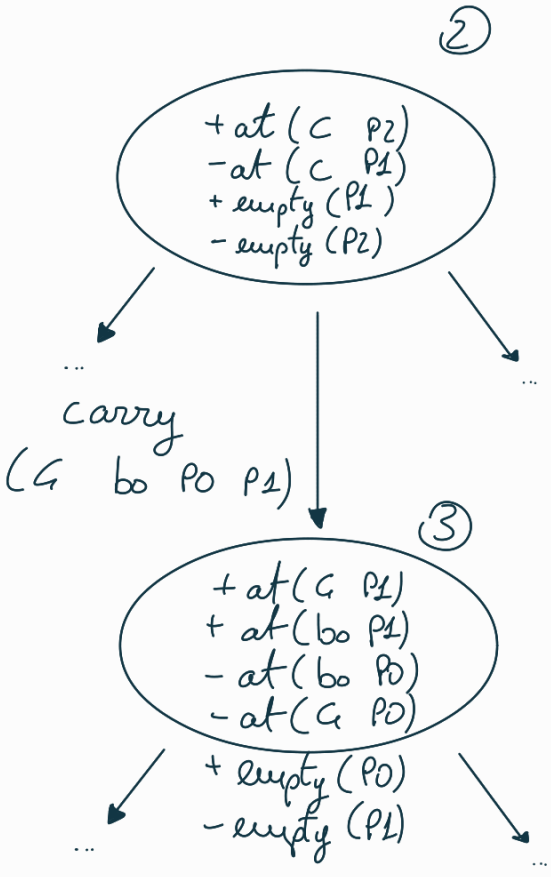
(b)

carry	c	ba	p6	p1
move	c	p1	p2	
carry	g	bo	p0	p1
carry	g	ba	p1	p3
move	c	p2	p1	
carry	c	bo	p1	p5



(c)




2
+ at (c p2)
- at (c p1)
+ empty (p1)
- empty (p2)
...
...
carry
(g bo p0 p1)
3
+ at (g p1)
+ at (bo p1)
- at (bo p0)
- at (g p0)
+ empty (p0)
- empty (p1)
...
...