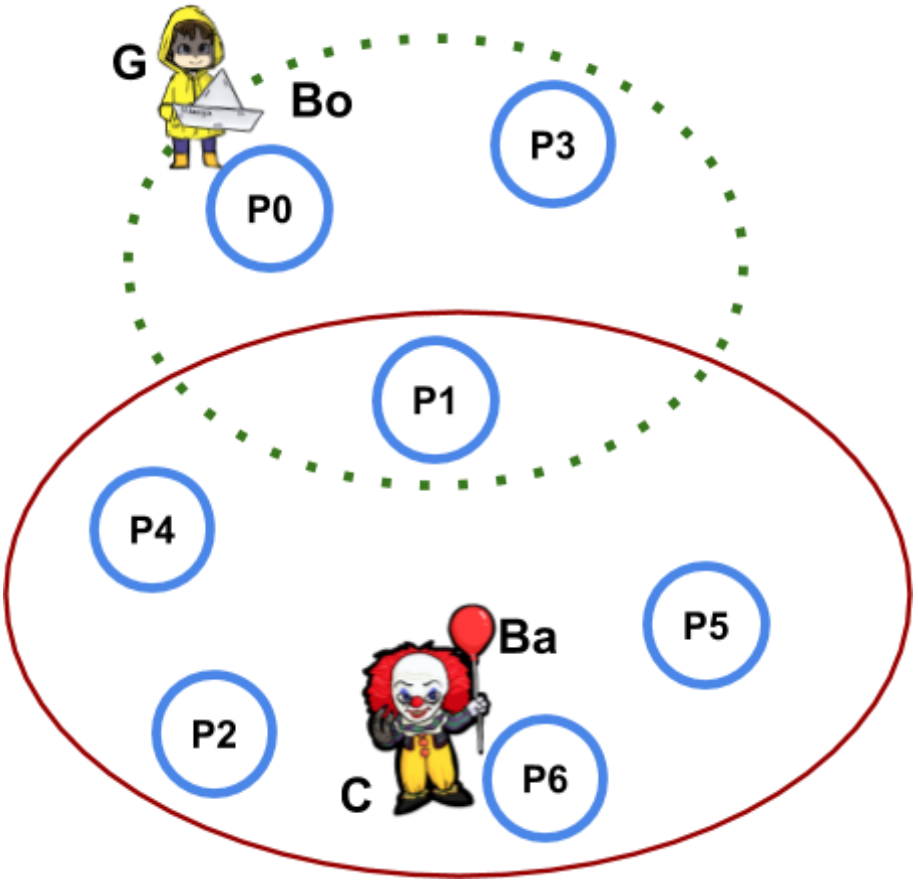
Track A

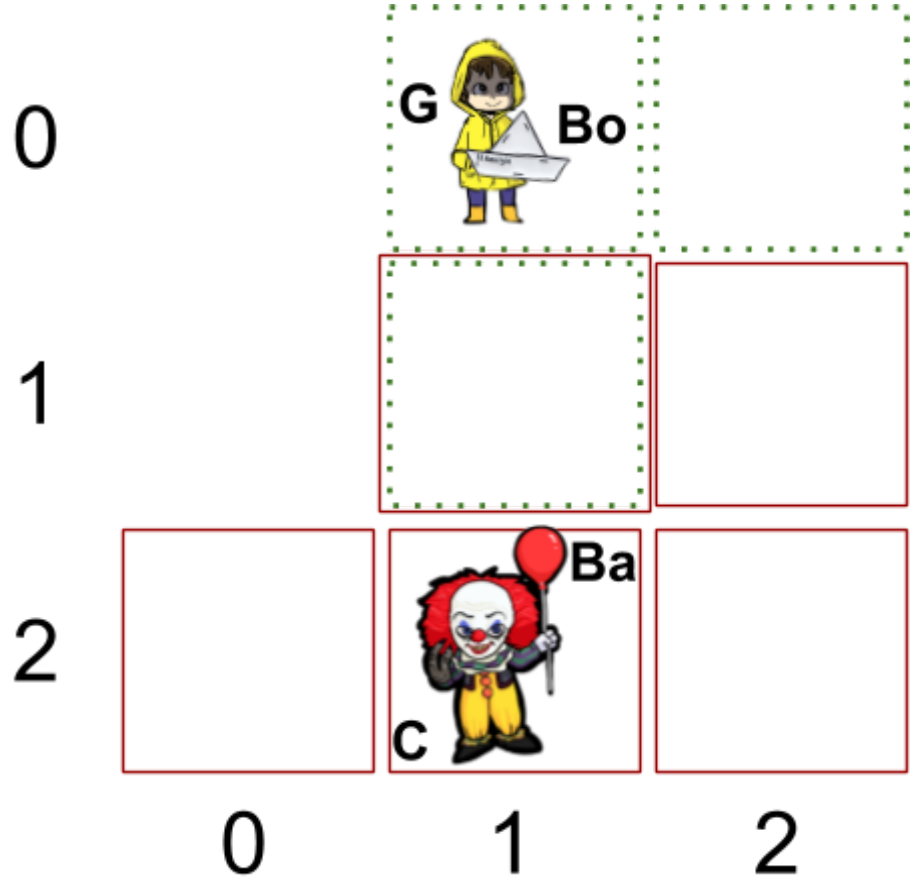
A clown **C** and Georgie **G** are tired of playing with their toys and they want to exchange them. G wants to play with the balloon **Ba** while, C wants to play with the boat **Bo**. Unfortunately, they are both very shy and they never want to be at the same place **Pi**. Hence, they have to find 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.



1. Model the problem in PDDL by defining the problem and domain file
2. Show a possible solution
3. Assuming a perfect heuristic, draw the first 5 steps of the tree generated by a forward search. Highlight actions on branches, and the current state on nodes.

Track B

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**. They are both very friendly and they want to be at the same cell **Cij** for exchanging them. Hence, they have to find a common place (for example Cell 1-1) where to drop and collect objects. G and C cannot hold two objects at the same time. Moreover, they cannot drop and collect objects if they are not in the same cell simultaneously.
The environment is depicted in the figure, G can only move within dotted cells; while C can only move within continuous cells. Cells of the same kind (i.e. {C-01, C-11, C-02} for the dotted set and {C-11, C-12, C-20, C-21, C-22} for the other) are connected only horizontally and vertically, in fact agents cannot move diagonally. The figure shows the initial state where G holds Bo and he is at C-01, and C holds Ba and he is at C-12. The goal state is represented by G at C-02 holding Ba and C at C-22 holding Bo.



1. Model the problem in PDDL by defining the problem and domain file
2. Show a possible solution
3. Assuming a perfect heuristic, draw the first 5 steps of the tree generated by a forward search. Highlight actions on branches, and the current state on nodes.

Track A - solution

```
(define (domain toy-exchange-domain)
 (:requirements :strips)
 (:predicates (at ?x ?y) (adjc ?x ?y) (adjg ?x ?y) (place ?x) (object ?x)
        (kid ?x) (clown ?x) (ballon ?x) (boat ?x) (hold ?x ?y) (free-hand ?x))

 (:action move-g
        :parameters (?from ?to)
        :precondition (and (at g ?from)
                (kid g)
                (place ?from) (place ?to)
                (adjg ?from ?to)
                (not (at c ?to)))
        :effect (and (not (at g ?from)) (at g ?to)))

 (:action move-c
        :parameters (?from ?to)
        :precondition (and (at c ?from)
                (clown c)
                (place ?from) (place ?to)
                (adjc ?from ?to)
                (not (at g ?to)))
        :effect (and (not (at c ?from)) (at c ?to)))

 (:action drop
        :parameters (?a ?o ?p)
        :precondition (and (at ?a p-1) (place ?p) (object ?o) (hold ?a ?o))
        :effect (and (at ?o ?p) (free-hand ?a) (not (hold ?a ?o))))

 (:action collect
        :parameters (?a ?o ?p)
        :precondition (and (at ?a p-1) (at ?o ?p) (place ?p) (object ?o) (not (hold ?a ?o))
                        (free-hand ?a))
        :effect (and (not (at ?o ?p)) (not (free-hand ?a)) (hold ?a ?o)))

)
```

```
(define (problem toy-exchange-problem)
  (:domain toy-exchange-domain)
  (:objects p-0 p-1 p-2 p-3 p-4 p-5 p-6 g c ba bo)
  (:init (adjg p-0 p-1) (adjg p-0 p-3) (adjg p-1 p-3)
         (adjg p-1 p-0) (adjg p-3 p-0) (adjg p-3 p-1)

    (adjc p-6 p-1) (adjc p-6 p-5) (adjc p-6 p-1)
    (adjc p-1 p-6) (adjc p-5 p-6) (adjc p-1 p-6)

     ; … other adjacencies

    (kid g) (clown c) (object ba) (object bo)
    (hold c ba) (hold g bo)
    (not (free-hand g)) (not (free-hand c))
    (at g p-0) (at c p-6)
    (place p-0) (place p-1) (place p-2) (place p-3)
         (place p-4) (place p-5) (place p-6)
  )
  (:goal (and (at g p-3) (at c p-5) (hold g ba) (hold c bo)))
)
```

Track B - solution

```
(define (domain toy-exchange-domain)
  (:requirements :strips)
  (:predicates (at ?x ?y) (adjc ?x ?y) (adjg ?x ?y) (place ?x) (object ?x)
         (kid ?x) (clown ?x) (ballon ?x) (boat ?x) (hold ?x ?y) (free-hand ?x))

  (:action move-g
         :parameters (?from ?to)
         :precondition (and (at g ?from)
                  (kid g)
                  (place ?from) (place ?to)
                  (adjg ?from ?to)))
         :effect (and (not (at g ?from)) (at g ?to)))

  (:action move-c
         :parameters (?from ?to)
         :precondition (and (at c ?from)
                  (clown c)
                  (place ?from) (place ?to)
                  (adjc ?from ?to))
         :effect (and (not (at c ?from)) (at c ?to)))
```

```
  (:action drop-g
        :parameters (?a ?o ?p)
        :precondition (and (at ?a c-11) (at g c-11) (kid ?a) (place ?p) (object ?o) (hold ?a ?o))
        :effect (and (at ?o ?p) (free-hand ?a) (not (hold ?a ?o))))

  (:action collect-g
        :parameters (?a ?o ?p)
        :precondition (and (at ?a p-1) (at ?o ?p) (at g c-11) (place ?p) (object ?o)
                           (not (hold ?a ?o)) (free-hand ?a))
        :effect (and (not (at ?o ?p)) (not (free-hand ?a)) (hold ?a ?o)))

)
```

Actions *drop-c* and *collect-c* are the same but for the agent performing the action.

```
(define (problem toy-exchange-problem)
  (:domain toy-exchange-domain)
  (:objects c-01 c-02 c-11 c-12 c-20 c-21 c-22 g c ba bo)
  (:init (adjg c-01 c-02) (adjg c-01 c-11) (adjg c-01 c-02)
        (adjg c-02 c-01) (adjg c-11 c-01) (adjg c-02 c-01)

    (adjc c-11 c-12) (adjc c-11 c-21) (adjc c-11 c-22)
    (adjc c-12 c-11) (adjc c-21 c-11) (adjc c-22 c-11)

     ; … other adjacencies

    (kid g) (clown c) (object ba) (object bo)
    (hold c ba) (hold g bo)
    (not (free-hand g)) (not (free-hand c))
    (at g c-01) (at c c-21)
    (place c-01) (place c-02) (place c-11) (place c-12)
        (place c-20) (place c-21) (place c-22)
  )
  (:goal (and (at g c-02) (at c c-22) (hold g ba) (hold c bo)))
)
```