School of Data and Computer Science, Sun Yat-sen University

# E06 FF Planner

18364066 Yanzuo Lu

October 12, 2020

## Contents

## 1 Examples

### 1.1 Spare Tire

domain_spare_tire.pddl

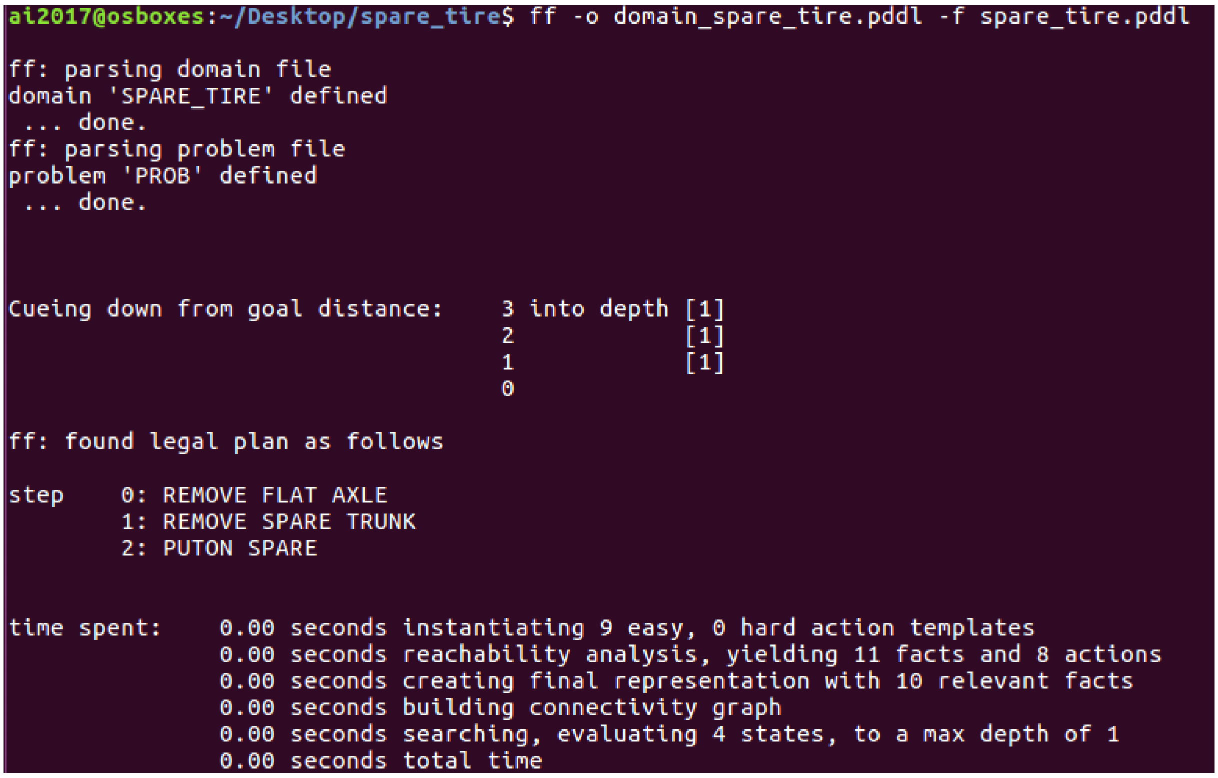
```
(define (domain spare_tire)
  (:requirements :strips :equality:typing)
  (:types physob location)
  (:predicates  (Tire ?x - physob)
                (at ?x - physob ?y - location))

(:action Remove
             :parameters (?x - physob ?y - location)
             :precondition (At ?x ?y)
             :effect (and (not (At ?x ?y)) (At ?x Ground)))

  (:action PutOn
             :parameters (?x - physob)
             :precondition (and (Tire ?x) (At ?x Ground)
                                (not (At Flat Axle)))
             :effect (and (not (At ?x Ground)) (At ?x Axle)))
  (:action LeaveOvernight
             :effect (and (not (At Spare Ground)) (not (At Spare Axle))
                          (not (At Spare Trunk)) (not (At Flat Ground))
                          (not (At Flat Axle)) (not (At Flat Trunk)) ))
 )
```

spare_tire.pddl

```
(define (problem prob)
 (:domain spare_tire)
 (:objects Flat Spare -physob Axle Trunk Ground - location)
 (:init (Tire Flat)(Tire Spare)(At Flat Axle)(At Spare Trunk))
 (:goal (At Spare Axle))
)
```



```
ai2017@osboxes:~/Desktop/spare_tire$ ff -o domain_spare_tire.pddl -f spare_tire.pddl

ff: parsing domain file
domain 'SPARE_TIRE' defined
 ... done.
ff: parsing problem file
problem 'PROB' defined
 ... done.



Cueing down from goal distance:    3 into depth [1]
                                   2            [1]
                                   1            [1]
                                   0

ff: found legal plan as follows

step    0: REMOVE FLAT AXLE
        1: REMOVE SPARE TRUNK
        2: PUTON SPARE


time spent:    0.00 seconds instantiating 9 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 11 facts and 8 actions
               0.00 seconds creating final representation with 10 relevant facts
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 4 states, to a max depth of 1
               0.00 seconds total time
```

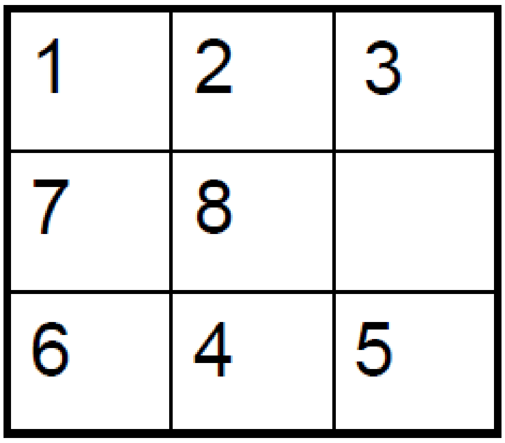
### 1.2 Briefcase World

Please refer to `pddl.pdf` at page 2. Please pay More attention to the usages of `forall` and `when`.

For more examples, please refer to `ff-domains.tgz` and `benchmarksV1.1.zip`. For more usages of FF planner, please refer to the documentation `pddl.pdf`.

## 2 Tasks

### 2.1 8-puzzle



|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 7 | 8 |   |
| 6 | 4 | 5 |

Please complete `domain_puzzle.pddl` and `puzzle.pddl` to solve the 8-puzzle problem.

domain_puzzle.pddl

```
(define (domain puzzle)
  (:requirements :strips :equality:typing)
  (:types num loc)
  (:predicates ())

(:action slide
             :parameters ()
             :precondition ()
             :effect ()
 )
)
```

domain_puzzle.pddl

```
(define (problem prob)
 (:domain puzzle)
 (:objects )
 (:init )
 (:goal ())
)
```

### 2.2 Blocks World

Planning in the blocks world is a traditional planning exercise, and you can recall what we have introduced in the theory course.
There are a collection of blocks: a block can be on the table, or on the top of another block.
There are three predicates:

- *clear(x)*: there is no block on top of block x;
- *on(x,y)*: block x is on the top of block y;
- *onTable(x)*: block x is on the table

There are two actions in this task:

- *move(x,y)*: move block x onto block y, provided that both x and y are clear;

- *moveToTable(x)*: move block x on to the table, provided that x is clear and x is not on the table;

Give initial state and goal state, find the actions change the initial state to the goal state.

In this task, please complete the file `domain_blocks.pddl` to solve the blocks world problem. You should know the usages of `forall` and `when`.

domain_blocks.pddl

```
(define (domain blocks)
  (:requirements :strips :typing:equality
                 :universal-preconditions
                 :conditional-effects)
  (:types physob)
  (:predicates
           (ontable ?x - physob)
           (clear ?x - physob)
           (on ?x ?y - physob))

  (:action move
            :parameters (?x ?y - physob)
            :precondition ()
            :effect ()
            )

  (:action moveToTable
            :parameters (?x - physob)
            :precondition ()
            :effect ( )
 )
```

blocks.pddl

```
(define (problem prob)
 (:domain blocks)
 (:objects A B C D E F - physob)
 (:init  (clear A)(on A B)(on B C)(ontable C)  (ontable D)
  (ontable F)(on E D)(clear E)(clear F)
)
 (:goal  (and (clear F)  (on F A)  (on A C)  (ontable C)(clear E)  (on E B)
         (on B D)  (ontable D)) )
 )
```

Please submit a file named E06_YourNumber.pdf, and send it to ai_2020@foxmail.com

## 3 Codes and Results

### 3.1 8-puzzle

domain_puzzle.pddl

```
;Header and description

(define (domain puzzle)

;remove requirements that are not needed
(:requirements :strips :typing :conditional-effects :equality)

(:types ;todo: enumerate types and their hierarchy here, e.g. car truck
   bus - vehicle
    num loc
)

; un-comment following line if constants are needed
;(:constants )

(:predicates ;todo: define predicates here
     (at ?x - num ?y - loc)
     (adjecent ?x - loc ?y - loc)
)


; (:functions ;todo: define numeric functions here
; )

;define actions here
(:action slide
    :parameters (?x - num ?y - loc ?z - loc)
    :precondition (and (at ?x ?y) (at num0 ?z) (adjecent ?y ?z))
    :effect (and (at ?x ?z) (at num0 ?y) (not (at ?x ?y)) (not (at num0
        ?z)))
)

)
```

puzzle.pddl

```
(define (problem prob) (:domain puzzle)
(:objects
    num0 num1 num2 num3 num4 num5 num6 num7 num8 - num
    loc1 loc2 loc3 loc4 loc5 loc6 loc7 loc8 loc0 - loc
)

(:init
    ;todo: put the initial state's facts and numeric values here
    (at num1 loc1) (at num2 loc2) (at num3 loc3)
    (at num7 loc4) (at num8 loc5) (at num0 loc6)
    (at num6 loc7) (at num4 loc8) (at num5 loc0)
    (adjecent loc1 loc2) (adjecent loc2 loc1)
    (adjecent loc1 loc4) (adjecent loc4 loc1)
    (adjecent loc2 loc3) (adjecent loc3 loc2)
    (adjecent loc2 loc5) (adjecent loc5 loc2)
    (adjecent loc3 loc6) (adjecent loc6 loc3)
    (adjecent loc4 loc5) (adjecent loc5 loc4)
    (adjecent loc4 loc7) (adjecent loc7 loc4)
    (adjecent loc5 loc6) (adjecent loc6 loc5)
    (adjecent loc5 loc8) (adjecent loc8 loc5)
    (adjecent loc6 loc0) (adjecent loc0 loc6)
    (adjecent loc7 loc8) (adjecent loc8 loc7)
    (adjecent loc8 loc0) (adjecent loc0 loc8)
)

(:goal (and
    ;todo: put the goal condition here
    (at num1 loc1) (at num2 loc2) (at num3 loc3)
    (at num4 loc4) (at num5 loc5) (at num6 loc6)
    (at num7 loc7) (at num8 loc8) (at num0 loc0)
))

;un-comment the following line if metric is needed
;(:metric minimize (???))
)
```

Plan.txt

```
(slide num5 loc0 loc6)
(slide num4 loc8 loc0)
(slide num8 loc5 loc8)
(slide num7 loc4 loc5)
(slide num6 loc7 loc4)
(slide num6 loc4 loc7)
(slide num7 loc5 loc4)
(slide num5 loc6 loc5)
(slide num4 loc0 loc6)
(slide num8 loc8 loc0)
(slide num5 loc5 loc8)
(slide num4 loc6 loc5)
(slide num8 loc0 loc6)
(slide num5 loc8 loc0)
```

```
(slide num6 loc7 loc8)
(slide num7 loc4 loc7)
(slide num4 loc5 loc4)
(slide num8 loc6 loc5)
(slide num5 loc0 loc6)
(slide num6 loc8 loc0)
(slide num8 loc5 loc8)
(slide num5 loc6 loc5)
(slide num6 loc0 loc6)
```

## 3.2 Blocks World

domain_blocks.pddl

```
;Header and description

(define (domain blocks)

;remove requirements that are not needed
(:requirements :strips :typing :conditional-effects :equality :
    universal-preconditions :negative-preconditions)

(:types ;todo: enumerate types and their hierarchy here, e.g. car truck
    bus - vehicle
     physob
)

; un-comment following line if constants are needed
;(:constants )

(:predicates ;todo: define predicates here
     (ontable ?x - physob)
     (clear ?x - physob)
     (on ?x ?y - physob)
)


; (:functions ;todo: define numeric functions here
; )

;define actions here
(:action move
    :parameters (?x ?y - physob)
    :precondition (and (clear ?x) (clear ?y) )
    :effect (and (on ?x ?y) (not (clear ?y))
        (when (ontable ?x) (not (ontable ?x)))
        (forall (?z - physob) (when (on ?x ?z) (and (not (on ?x ?z)) (
            clear ?z))))
    )
)

(:action moveToTable
    :parameters (?x - physob)
    :precondition (and (clear ?x) (not (ontable ?x)))
    :effect (and (not (clear ?x)) (ontable ?x)
        (forall (?z - physob) (when (on ?x ?z) (and (not (on ?x ?z)) (
            clear ?z))))
    )
)


)
```

blocks.pddl

```
(define (problem prob) (:domain blocks)
(:objects
    A B C D E F - physob
)

(:init
    ;todo: put the initial state's facts and numeric values here
    (clear A) (on A B) (on B C) (ontable C) (ontable D)
    (ontable F) (on E D) (clear E) (clear F)
)

(:goal (and
    ;todo: put the goal condition here
    (clear F) (on F A) (on A C) (ontable C) (clear E)
    (on E B) (on B D) (ontable D)
))

;un-comment the following line if metric is needed
;(:metric minimize (???))
)
```

Plan.txt

```
(move a f)
(move b e)
(move a c)
(move b a)
(move e f)
(move b d)
(move e b)
(move f a)
```

### 3.3 Results

本次实验非常简单，重点在于对 PPDL 语法的掌握和理解，问题本身的一阶逻辑并不复杂，需要注意的是 FF 规划器并不能生成最优的方案，例如在 Puzzle 问题中就出现了连续拨动数字 6 的情况，经过手动验证两个问题的 Plan 都是正确的。（注：代码中的注释由 Vscode 插件自动生成）