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

# E06 FF Planner

Ze He

October 13, 2020

## Contents

# 1 Examples

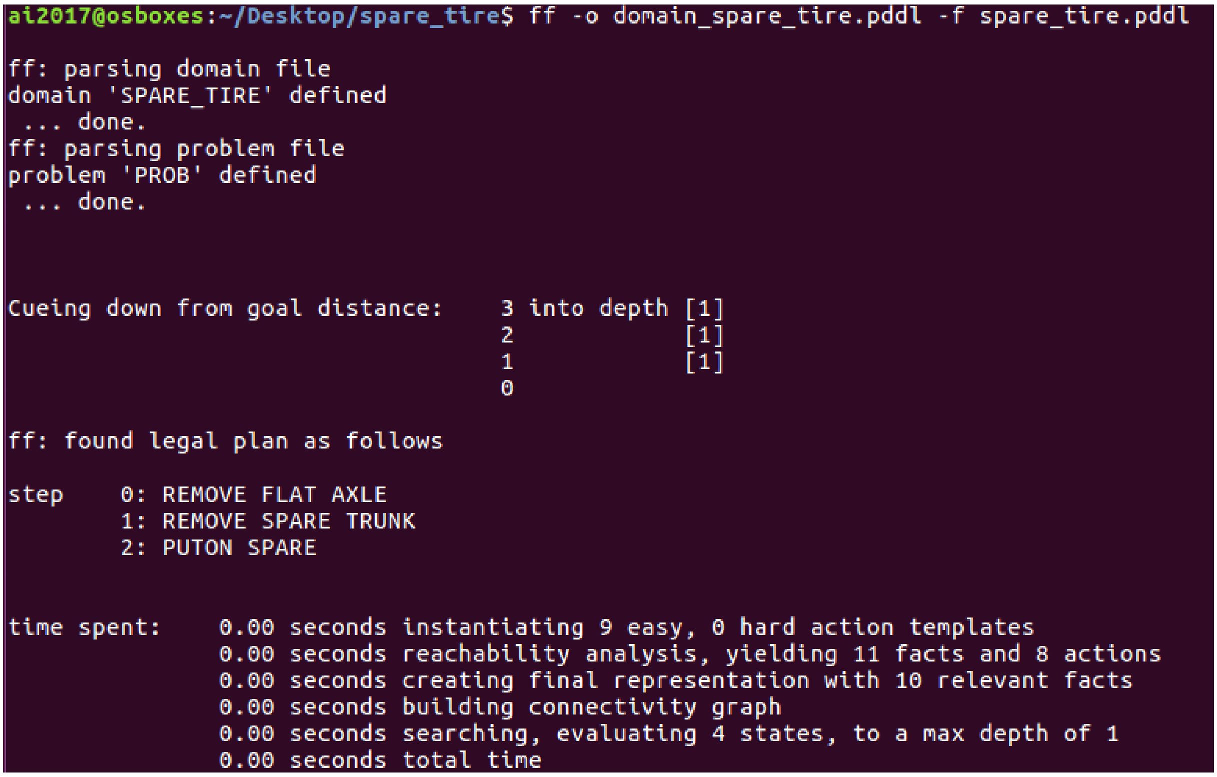
## 1.1 Spare Tire

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

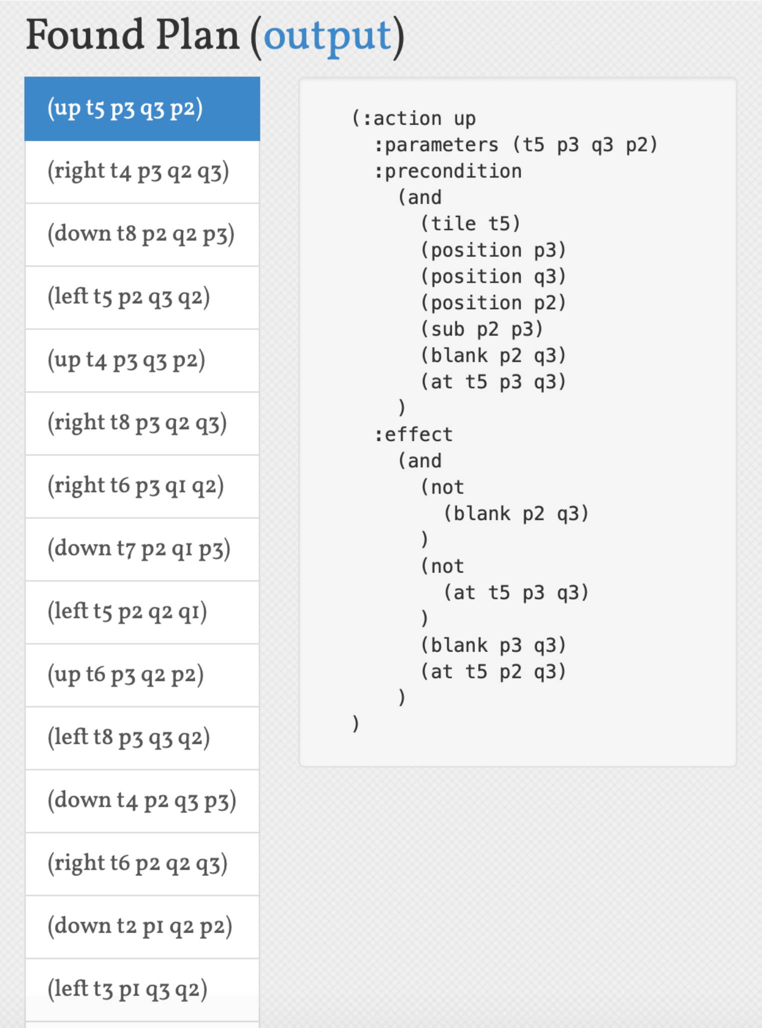
```
(define (domain puzzle)
    (:requirements
        :strips :equality :typing)
    (:predicates
        (tile ?x) (position ?x)
        (at ?t ?x ?y) (blank ?x ?y)
        (add ?new ?pre) (sub ?new ?pre))

     (:action up
     :parameters (?t ?x ?y ?x_new)
     :precondition (
         and (tile ?t) (position ?x) (position ?y) (position ?x_new)
                     (sub ?x_new ?x) (blank ?x_new ?y) (at ?t ?x ?y))
     :effect (
         and (not (blank ?x_new ?y)) (not (at ?t ?x ?y))
                     (blank ?x ?y) (at ?t ?x_new ?y)))

    (:action down
    :parameters (?t ?x ?y ?x_new)
    :precondition (
        and (tile ?t) (position ?x) (position ?y) (position ?x_new)
                    (add ?x_new ?x) (blank ?x_new ?y) (at ?t ?x ?y))
    :effect (and (not (blank ?x_new ?y)) (not (at ?t ?x ?y))
                    (blank ?x ?y) (at ?t ?x_new ?y)))

    (:action left
    :parameters (?t ?x ?y ?y_new)
    :precondition (
        and (tile ?t) (position ?x) (position ?y) (position ?y_new)
                (sub ?y_new ?y) (blank ?x ?y_new) (at ?t ?x ?y))
    :effect (
        and (not (blank ?x ?y_new)) (not (at ?t ?x ?y))
                (blank ?x ?y) (at ?t ?x ?y_new)))

    (:action right
    :parameters (?t ?x ?y ?y_new)
    :precondition (
        and (tile ?t) (position ?x) (position ?y) (position ?y_new)
                (add ?y_new ?y) (blank ?x ?y_new) (at ?t ?x ?y))
    :effect (
        and (not (blank ?x ?y_new)) (not (at ?t ?x ?y))
                (blank ?x ?y) (at ?t ?x ?y_new)))
)
```

domain_puzzle.pddl

```
(define (problem prob)
    (:domain puzzle)
    (:objects t1 t2 t3 t4 t5 t6 t7 t8 p1 p2 p3 q1 q2 q3)
    (:init (tile t1) (tile t2) (tile t3) (tile t4)
           (tile t5) (tile t6) (tile t7) (tile t8)
           (position p1) (position p2)
           (position p3) (position q1)
           (position q2) (position q3)
```

```
            (blank p2 q3)
            (at t1 p1 q1) (at t2 p1 q2) (at t3 p1 q3) (at t7 p2 q1)
            (at t8 p2 q2) (at t6 p3 q1) (at t4 p3 q2) (at t5 p3 q3)
            (add p2 p1) (add p3 p2) (sub p1 p2) (sub p2 p3)
            (add q2 q1) (add q3 q2) (sub q1 q2) (sub q2 q3) )
    (:goal (
        and (at t1 p1 q1) (at t2 p1 q2) (at t3 p1 q3) (at t4 p2 q1)
             (at t5 p2 q2) (at t6 p2 q3) (at t7 p3 q1) (at t8 p3 q2)
             (blank p3 q3) ))
)
```

Results:



Found Plan (output)

(up t5 p3 q3 p2)
(right t4 p3 q2 q3)
(down t8 p2 q2 p3)
(left t5 p2 q3 q2)
(up t4 p3 q3 p2)
(right t8 p3 q2 q3)
(right t6 p3 q1 q2)
(down t7 p2 q1 p3)
(left t5 p2 q2 q1)
(up t6 p3 q2 p2)
(left t8 p3 q3 q2)
(down t4 p2 q3 p3)
(right t6 p2 q2 q3)
(down t2 p1 q2 p2)
(left t3 p1 q3 q2)

```
(:action up
  :parameters (t5 p3 q3 p2)
  :precondition
    (and
      (tile t5)
      (position p3)
      (position q3)
      (position p2)
      (sub p2 p3)
      (blank p2 q3)
      (at t5 p3 q3)
    )
  :effect
    (and
      (not
        (blank p2 q3)
      )
      (not
        (at t5 p3 q3)
      )
      (blank p3 q3)
      (at t5 p2 q3)
    )
)
```

Part of the result on the website

All results:

result1

```
(up t5 p3 q3 p2)
(right t4 p3 q2 q3)
(down t8 p2 q2 p3)
(left t5 p2 q3 q2)
(up t4 p3 q3 p2)
(right t8 p3 q2 q3)
(right t6 p3 q1 q2)
(down t7 p2 q1 p3)
(left t5 p2 q2 q1)
(up t6 p3 q2 p2)
(left t8 p3 q3 q2)
(down t4 p2 q3 p3)
(right t6 p2 q2 q3)
(down t2 p1 q2 p2)
(left t3 p1 q3 q2)
(up t6 p2 q3 p1)
(up t4 p3 q3 p2)
(right t8 p3 q2 q3)
(down t2 p2 q2 p3)
(left t4 p2 q3 q2)
(down t6 p1 q3 p2)
(right t3 p1 q2 q3)
(up t4 p2 q2 p1)
(right t5 p2 q1 q2)
(down t1 p1 q1 p2)
(left t4 p1 q2 q1)
(up t5 p2 q2 p1)
(up t2 p3 q2 p2)
(right t7 p3 q1 q2)
(down t1 p2 q1 p3)
(down t4 p1 q1 p2)
```

```
(left t5 p1 q2 q1)
(up t2 p2 q2 p1)
(right t4 p2 q1 q2)
(up t1 p3 q1 p2)
(left t7 p3 q2 q1)
(down t4 p2 q2 p3)
(right t1 p2 q1 q2)
(down t5 p1 q1 p2)
(left t2 p1 q2 q1)
(up t1 p2 q2 p1)
(right t5 p2 q1 q2)
(down t2 p1 q1 p2)
(left t1 p1 q2 q1)
(up t5 p2 q2 p1)
(right t2 p2 q1 q2)
(up t7 p3 q1 p2)
(left t4 p3 q2 q1)
(down t2 p2 q2 p3)
(right t7 p2 q1 q2)
(up t4 p3 q1 p2)
(left t2 p3 q2 q1)
(down t7 p2 q2 p3)
(down t5 p1 q2 p2)
(right t1 p1 q1 q2)
(up t4 p2 q1 p1)
(up t2 p3 q1 p2)
(left t7 p3 q2 q1)
(down t5 p2 q2 p3)
(right t2 p2 q1 q2)
(down t4 p1 q1 p2)
(left t1 p1 q2 q1)
(up t2 p2 q2 p1)
(up t5 p3 q2 p2)
```

```
(left t8 p3 q3 q2)
```

## 3.2 Blocks World

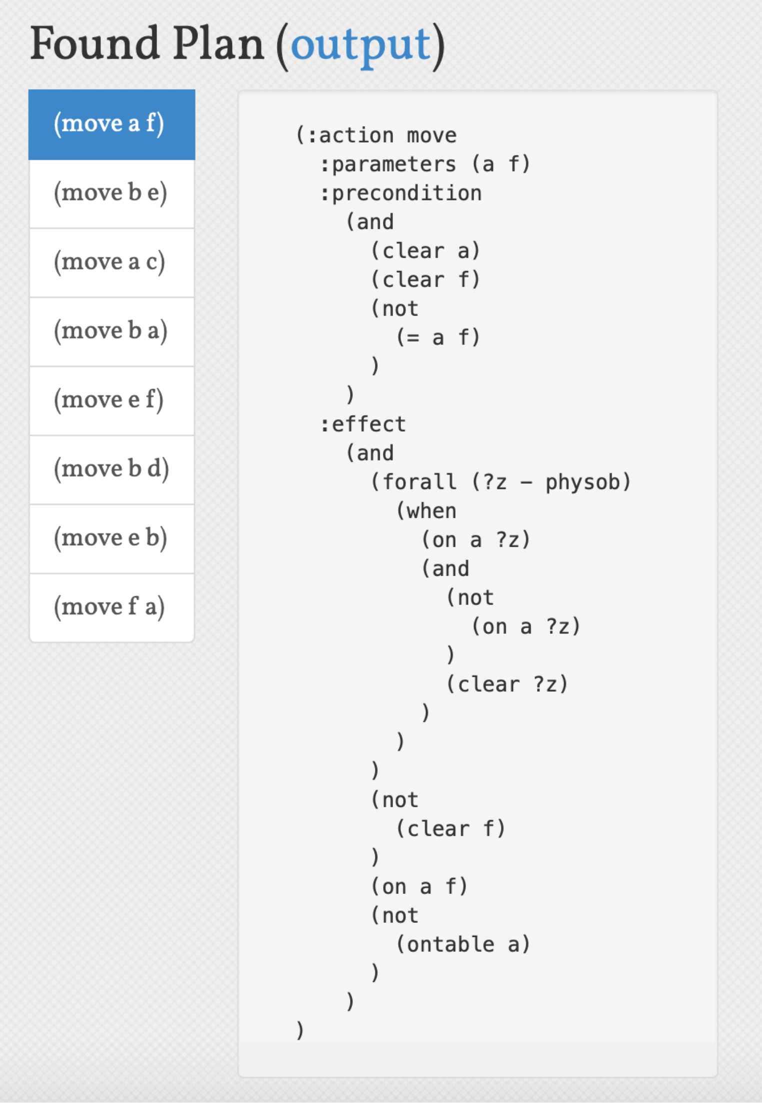
domain_blocks.pddl

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

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

    (:action moveToTable
        :parameters (?x - physob)
        :precondition(
            and (clear ?x) (not (ontable ?x)))
        :effect( and (ontable ?x) ))
)
```

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)) )
 )
```

Results:



Found Plan (output)

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

```
(:action move
  :parameters (a f)
  :precondition
    (and
      (clear a)
      (clear f)
      (not
        (= a f)
      )
    )
  :effect
    (and
      (forall (?z - physob)
        (when
          (on a ?z)
          (and
            (not
              (on a ?z)
            )
            (clear ?z)
          )
        )
      )
      (not
        (clear f)
      )
      (on a f)
      (not
        (ontable a)
      )
    )
)
```