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

E07 FF Planner

17341015 Hongzheng Chen

October 19, 2019

Contents

1 Examples

1.1 Spare Tire

domain_spare_tire.pddl

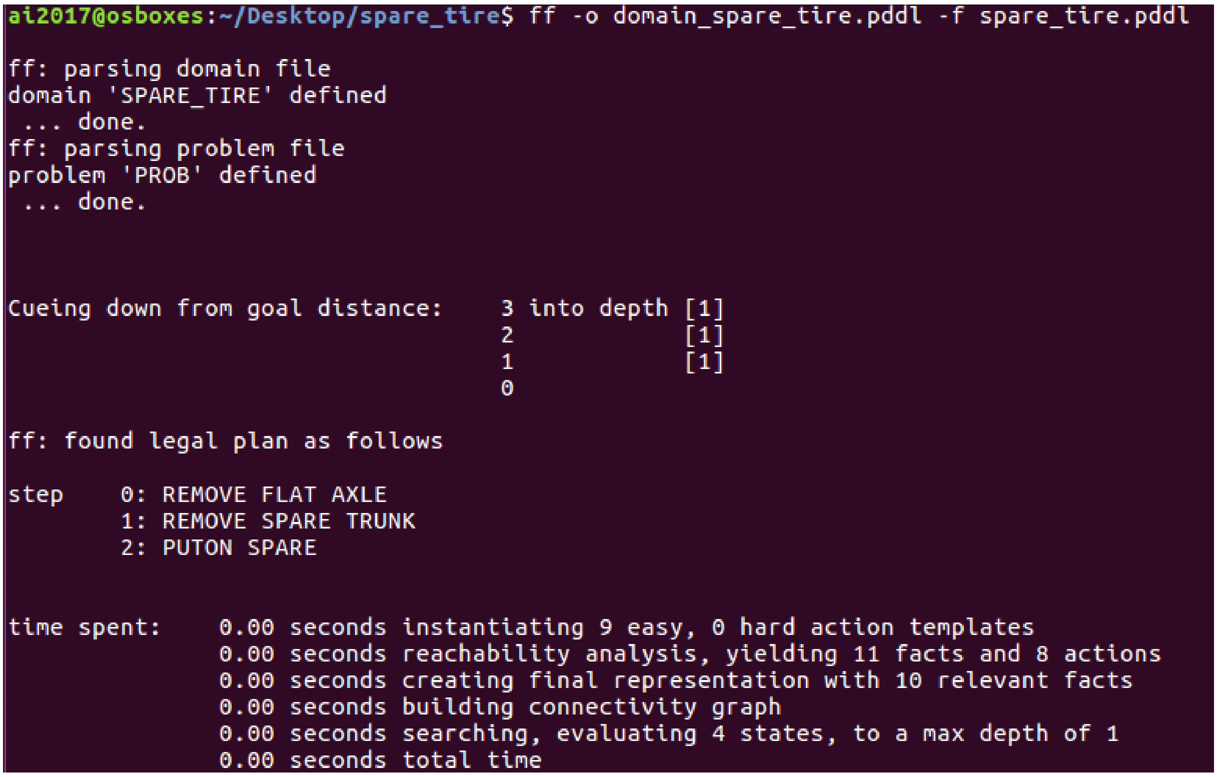
```
(define (domain spare_tire)
  (:requirements :strips :equality:typing)
  (:types physob location)
  (:predicates (Tire ?x - physob)
       (at ?x - physob ?y - location))

(: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

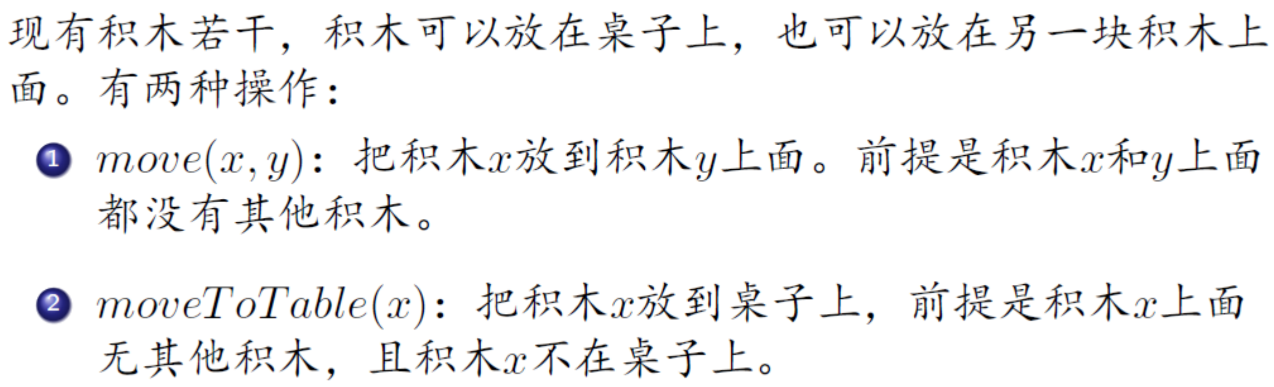
```
(define (domain puzzle)
  (:requirements :strips :equality:typing)
  (:types num loc)
  (:predicates ())

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

domain_puzzle.pddl

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

2.2 Blocks World



现有积木若干，积木可以放在桌子上，也可以放在另一块积木上面。有两种操作：

❶ $move(x, y)$：把积木x放到积木y上面。前提是积木x和y上面都没有其他积木。

❷ $moveToTable(x)$：把积木x放到桌子上，前提是积木x上面无其他积木，且积木x不在桌子上。

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 E07_YourNumber.pdf, and send it to ai_201901@foxmail.com

3 Codes and Results

3.1 8-Puzzle

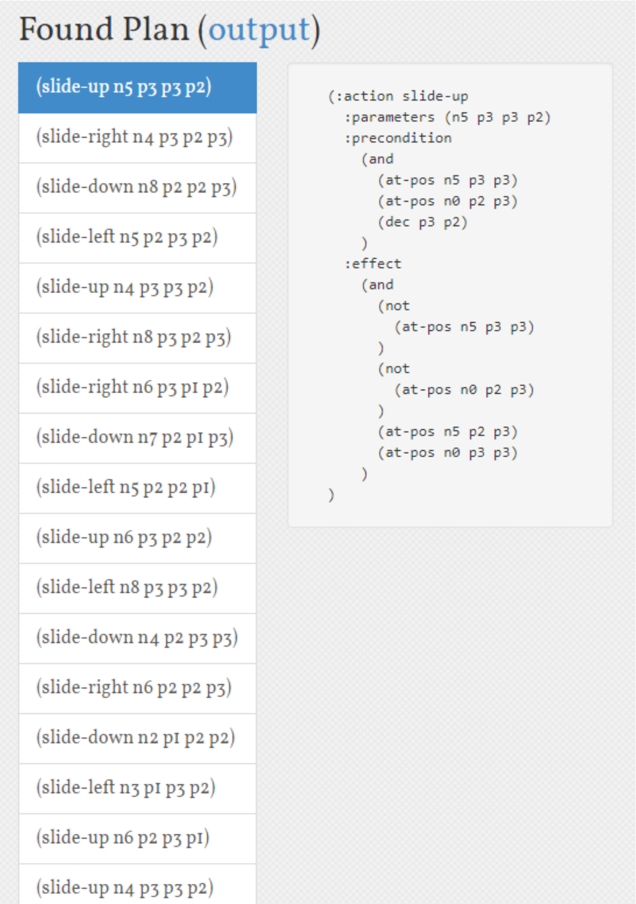
The code below is `domain_puzzle.pddl`, which leverages four slide action to describe the movements.

```
(define (domain puzzle)
(:requirements :strips :typing)
(:types num loc)
(:predicates
    (at-pos ?n ?x ?y)
    (inc ?p ?p1)
    (dec ?p ?p1))
(:action slide-up
    :parameters (?n ?x ?y ?x1)
    :precondition (and (at-pos ?n ?x ?y) (at-pos n0 ?x1 ?y) (dec ?x ?x1))
    :effect (and (not (at-pos ?n ?x ?y)) (not (at-pos n0 ?x1 ?y))
                (at-pos ?n ?x1 ?y) (at-pos n0 ?x ?y))
)
(:action slide-down
    :parameters (?n ?x ?y ?x1)
    :precondition (and (at-pos ?n ?x ?y) (at-pos n0 ?x1 ?y) (inc ?x ?x1))
    :effect (and (not (at-pos ?n ?x ?y)) (not (at-pos n0 ?x1 ?y))
                (at-pos ?n ?x1 ?y) (at-pos n0 ?x ?y))
)
(:action slide-left
    :parameters (?n ?x ?y ?y1)
    :precondition (and (at-pos ?n ?x ?y) (at-pos n0 ?x ?y1) (dec ?y ?y1))
    :effect (and (not (at-pos ?n ?x ?y)) (not (at-pos n0 ?x ?y1))
                (at-pos ?n ?x ?y1) (at-pos n0 ?x ?y))
)
(:action slide-right
    :parameters (?n ?x ?y ?y1)
    :precondition (and (at-pos ?n ?x ?y) (at-pos n0 ?x ?y1) (inc ?y ?y1))
    :effect (and (not (at-pos ?n ?x ?y)) (not (at-pos n0 ?x ?y1))
                (at-pos ?n ?x ?y1) (at-pos n0 ?x ?y))
)
)
```

Below is `puzzle.pddl`.

```
(define (problem prob)
  (:domain puzzle)
  (:objects n0 n1 n2 n3 n4 n5 n6 n7 n8 p1 p2 p3)
  (:init
      (inc p1 p2)
      (inc p2 p3)
      (dec p3 p2)
      (dec p2 p1)
      (at-pos n1 p1 p1)
      (at-pos n2 p1 p2)
      (at-pos n3 p1 p3)
      (at-pos n7 p2 p1)
      (at-pos n8 p2 p2)
      (at-pos n0 p2 p3)
      (at-pos n6 p3 p1)
      (at-pos n4 p3 p2)
      (at-pos n5 p3 p3)
  )
  (:goal (and
      (at-pos n1 p1 p1)
      (at-pos n2 p1 p2)
      (at-pos n3 p1 p3)
      (at-pos n4 p2 p1)
      (at-pos n5 p2 p2)
      (at-pos n6 p2 p3)
      (at-pos n7 p3 p1)
      (at-pos n8 p3 p2)
      (at-pos n0 p3 p3)
  ))
)
```

The running result is shown below. For all the actions, please refer to the attached `result1.txt`.



Found Plan (output)

- (slide-up n5 p3 p3 p2)
- (slide-right n4 p3 p2 p3)
- (slide-down n8 p2 p2 p3)
- (slide-left n5 p2 p3 p2)
- (slide-up n4 p3 p3 p2)
- (slide-right n8 p3 p2 p3)
- (slide-right n6 p3 p1 p2)
- (slide-down n7 p2 p1 p3)
- (slide-left n5 p2 p2 p1)
- (slide-up n6 p3 p2 p2)
- (slide-left n8 p3 p3 p2)
- (slide-down n4 p2 p3 p3)
- (slide-right n6 p2 p2 p3)
- (slide-down n2 p1 p2 p2)
- (slide-left n3 p1 p3 p2)
- (slide-up n6 p2 p3 p1)
- (slide-up n4 p3 p3 p2)

```
(:action slide-up
  :parameters (n5 p3 p3 p2)
  :precondition
    (and
      (at-pos n5 p3 p3)
      (at-pos n0 p2 p3)
      (dec p3 p2)
    )
  :effect
    (and
      (not
        (at-pos n5 p3 p3)
      )
      (not
        (at-pos n0 p2 p3)
      )
      (at-pos n5 p2 p3)
      (at-pos n0 p3 p3)
    )
)
```

3.2 Block Worlds

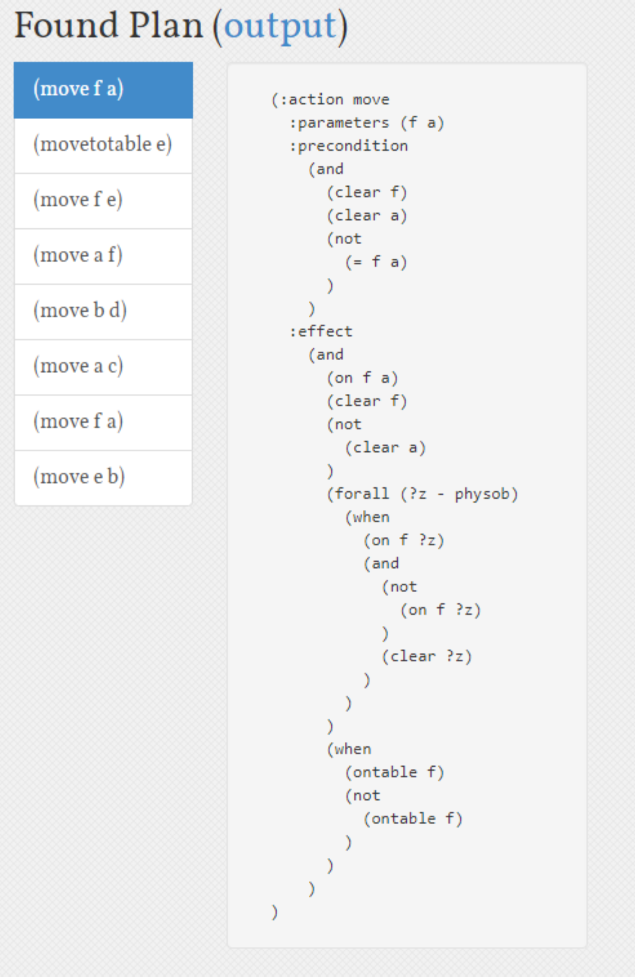
Below is `domain_blocks`, which leverages `forall` and `when` grammas to specify the blocks below which to be moved.

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

Below shows `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))
)
)
```

The result is shown below.



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

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