

E07 FF Planner

17341189 姚森舰

2019 年 10 月 19 日

目录

1	Examples	2
1.1	Spare Tire	2
1.2	Briefcase World	2
2	Tasks	3
2.1	8-puzzle	3
2.2	Blocks World	4
3	Codes and Results	5
3.1	8-puzzle	5
3.2	Blocks World	8

1 Examples

1.1 Spare Tire

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

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

11
  (:action PutOn
13    :parameters (?x - physob)
    :precondition (and (Tire ?x) (At ?x Ground)
15                      (not (At Flat Axle)))
    :effect (and (not (At ?x Ground)) (At ?x Axle)))
17  (:action LeaveOvernight
    :effect (and (not (At Spare Ground)) (not (At Spare Axle))
19                (not (At Spare Trunk)) (not (At Flat Ground))
                (not (At Flat Axle)) (not (At Flat Trunk)) ))
21 )
```

domain_spare_tire.pddl

```
(define (problem prob)
2  (:domain spare_tire)
  (:objects Flat Spare -physob Axle Trunk Ground - location)
4  (:init (Tire Flat)(Tire Spare)(At Flat Axle)(At Spare Trunk))
  (:goal (At Spare Axle))
6 )
```

spare_tire.pddl

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`.

```

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

```

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.

```

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

6  (:action slide
      :parameters ()

```

```

8           :precondition ()
           :effect ()
10 )
)

```

domain_puzzle.pddl

```

1 (define (problem prob)
  (:domain puzzle)
3  (:objects )
  (:init )
5  (:goal ()))
)

```

domain_puzzle.pddl

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`.

```

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

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

```

```

15         )
17     (:action moveToTable
18         :parameters (?x – physob)
19         :precondition ()
20         :effect ( )
21 )

```

domain_blocks.pddl

```

(define (problem prob)
2  (:domain blocks)
  (:objects A B C D E F – physob)
4  (:init (clear A)(on A B)(on B C)(ontable C) (ontable D)
  (ontable F)(on E D)(clear E)(clear F)
6  )
  (:goal (and (clear F) (on F A) (on A C) (ontable C)(clear E) (on E B)
8      (on B D) (ontable D)) )
  )

```

blocks.pddl

Please submit a file named E07_YourNumber.pdf, and send it to ai_201901@foxmail.com

3 Codes and Results

3.1 8-puzzle

```

1 (define (domain puzzle)
  (:requirements :strips :equality :typing)
3  (:types num loc)
  (:predicates (at ?x – num ?y – loc)
5      (adj ?m – loc ?n – loc))

7  (:action slide
    :parameters (?mov_num – num ?cur_loc – loc ?blank_loc – loc)
9    :precondition (and
        (at ?mov_num ?cur_loc)
11       (at n0 ?blank_loc)
        (adj ?cur_loc ?blank_loc)
13       )
    :effect (and
15       (at ?mov_num ?blank_loc)
        (not (at ?mov_num ?cur_loc))
17       (at n0 ?cur_loc)

```

```

19         (not (at n0 ?blank_loc))
21     )

```

domain_puzzle.pddl

```

1 (define (problem prob)
  (:domain puzzle)
3  (:objects n0 n1 n2 n3 n4 n5 n6 n7 n8 - num
        11 12 13 14 15 16 17 18 19 - loc)
5
  (:init (at n1 11) (at n2 12) (at n3 13)
6         (at n7 14) (at n8 15) (at n0 16)
7         (at n6 17) (at n4 18) (at n5 19)
8
9         (adj 11 12) (adj 11 14)
11        (adj 12 11) (adj 12 13) (adj 12 15)
13        (adj 13 12) (adj 13 16)
15        (adj 14 11) (adj 14 15) (adj 14 17)
17        (adj 15 14) (adj 15 12) (adj 15 16) (adj 15 18)
19        (adj 16 13) (adj 16 15) (adj 16 19)
21        (adj 17 14) (adj 17 18)
23        (adj 18 17) (adj 18 19) (adj 18 15)
25        (adj 19 16) (adj 19 18)
27    )
  (:goal (and
        (at n1 11) (at n2 12) (at n3 13)
        (at n4 14) (at n5 15) (at n6 16)
        (at n7 17) (at n8 18) (at n0 19)
    ))
)

```

problem_puzzle.pddl

表示将要移动的方块

The diagram illustrates a sequence of 30 steps for solving a 3x3x3 Rubik's cube, organized into four groups of seven steps each. Each step shows a 3x3 grid with one block highlighted in orange, indicating the block to be moved. The sequence starts with a scrambled cube and ends with a solved cube.

Group 1 (Steps 1-7):

- Step 1: Initial state. Orange block: 5.
- Step 2: Orange block: 8.
- Step 3: Orange block: 5.
- Step 4: Orange block: 4.
- Step 5: Orange block: 8.
- Step 6: Orange block: 6.
- Step 7: Orange block: 7.

Group 2 (Steps 8-14):

- Step 8: Orange block: 6.
- Step 9: Orange block: 8.
- Step 10: Orange block: 4.
- Step 11: Orange block: 6.
- Step 12: Orange block: 2.
- Step 13: Orange block: 3.
- Step 14: Orange block: 6.

Group 3 (Steps 15-21):

- Step 15: Orange block: 8.
- Step 16: Orange block: 2.
- Step 17: Orange block: 4.
- Step 18: Orange block: 6.
- Step 19: Orange block: 3.
- Step 20: Orange block: 4.
- Step 21: Orange block: 5.

Group 4 (Steps 22-28):

- Step 22: Orange block: 4.
- Step 23: Orange block: 5.
- Step 24: Orange block: 2.
- Step 25: Orange block: 7.
- Step 26: Orange block: 1.
- Step 27: Orange block: 5.
- Step 28: Orange block: 2.

Final Step (Step 29):

- Step 29: Final state. Orange block: 1.

7

Found Plan (output)

(slide n4 l4 l7)

(slide n5 l1 l4)

(slide n1 l2 l1)

(slide n2 l3 l2)

(slide n3 l6 l3)

(slide n6 l5 l6)

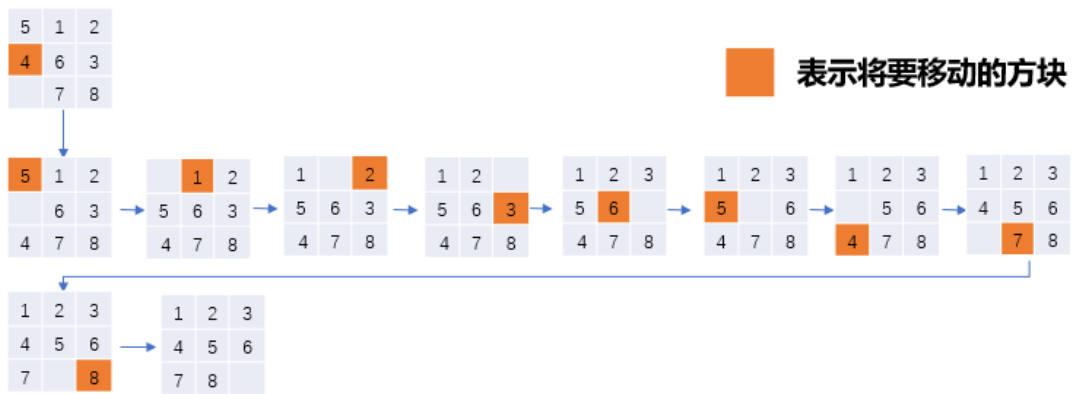
(slide n5 l4 l5)

(slide n4 l7 l4)

(slide n7 l8 l7)

(slide n8 l9 l8)

```
(:action slide
:parameters (n4 l4 l7)
:precondition
  (and
    (at n4 l4)
    (at n0 l7)
    (adj l4 l7)
  )
:effect
  (and
    (at n4 l7)
    (not
      (at n4 l4)
    )
    (at n0 l4)
    (not
      (at n0 l7)
    )
  )
)
```



3.2 Blocks World

```
1 (define (domain blocks)
  (:requirements :strips :typing :equality
3    :universal-preconditions :conditional-effects)
  (:types physob)
5  (:predicates (ontable ?x - physob)
              (clear ?x - physob)
7              (on ?x ?y - physob)))
9  (:action move
      :parameters (?x ?y - physob)
```



```

11      :precondition (and
12                    (clear ?x)
13                    (clear ?y)
14                    )
15
16      :effect (and
17              (
18                forall (?z - physob)
19                  (when (on ?x ?z)
20                    (and (clear ?z) (not (on ?x ?z)))
21                  )
22              )
23              (not (clear ?y))
24              (on ?x ?y)
25            )
26          )
27
28      (:action moveToTable
29        :parameters (?x - physob)
30        :precondition (and
31                      (clear ?x)
32                      (not (ontable ?x))
33                    )
34        :effect (and
35              (
36                forall (?z - physob)
37                  (when (on ?x ?z)
38                    (and (clear ?z) (not (on ?x ?z) ) )
39                  )
40              )
41              (ontable ?x)
42            )
43          )
44    )

```

domain_block.pddl

```

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

```

10

)

)

problem_block.pddl

以下是运行结果：

Found Plan (output)

(movetotable a)

(movetotable e)

(move b d)

(move a c)

(move fa)

(move e b)

```

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

```

经验证结果正确。这里面需要注意的是如果 x 在 z 上，当 x 从 z 上移走时，其中一个后果是 z 上面就 clear 了，不加这句话就会出错。