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

E07 FF Planner

17341137 Zhenpeng Song

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
```

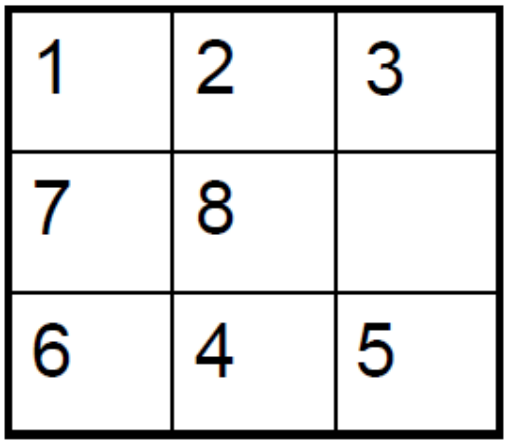
1.2 Briefcase World

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



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：

1. $move(x, y)$：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。
2. $moveToTable(x)$：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

The codes and planning results are also included in the .zip file.

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 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)))
    (:action up
    :parameters (?t ?x ?y ?x_new)
    :precondition (
```

```
28          and (tile ?t) (position ?x) (position ?y) (position ?x_new)
29              (sub ?x_new ?x) (blank ?x_new ?y) (at ?t ?x ?y))
30      :effect (
31          and (not (blank ?x_new ?y)) (not (at ?t ?x ?y))
32              (blank ?x ?y) (at ?t ?x_new ?y)))
33      (:action down
34      :parameters (?t ?x ?y ?x_new)
35      :precondition (and
36              (tile ?t) (position ?x) (position ?y) (position ?x_new)
37              (add ?x_new ?x) (blank ?x_new ?y) (at ?t ?x ?y))
38      :effect (and (not (blank ?x_new ?y)) (not (at ?t ?x ?y))
39           (blank ?x ?y) (at ?t ?x_new ?y))))
```

puzzle.pddl

```
1  (define (problem prob)
2      (:domain puzzle)
3      (:objects t1 t2 t3 t4 t5 t6 t7 t8 p1 p2 p3 q1 q2 q3)
4      (:init (tile t1) (tile t2) (tile t3) (tile t4)
5              (tile t5) (tile t6) (tile t7) (tile t8)
6              (position p1) (position p2)
7              (position p3) (position q1)
8              (position q2) (position q3)
9              (blank p2 q3)
10             (at t1 p1 q1) (at t2 p1 q2) (at t3 p1 q3) (at t7 p2 q1)
11             (at t8 p2 q2) (at t6 p3 q1) (at t4 p3 q2) (at t5 p3 q3)
12             (add p2 p1) (add p3 p2) (sub p1 p2) (sub p2 p3)
13             (add q2 q1) (add q3 q2) (sub q1 q2) (sub q2 q3))
14     (:goal (
15         and (at t1 p1 q1) (at t2 p1 q2) (at t3 p1 q3) (at t4 p2 q1)
16             (at t5 p2 q2) (at t6 p2 q3) (at t7 p3 q1) (at t8 p3 q2)
17             (blank p3 q3)
18     )))
```

Result:

1. (up t5 p3 q3 p2)
2. (right t4 p3 q2 q3)
3. ...
4. (up t5 p3 q2 p2)
5. (left t8 p3 q3 q2)

In other words:

U 5⟶R 4⟶ D 8⟶L 5⟶ U 4⟶R 8⟶ R 6⟶D 7⟶ L 5⟶U 6⟶ L 8⟶D 4⟶ R 6⟶D 2⟶ L 3⟶U 6⟶ U 4⟶R 8⟶ D 2⟶L 4⟶ D 6⟶R 3⟶ U 4⟶R 5⟶ D 1⟶L 4⟶ U 5⟶U 2⟶ R 7⟶D 1⟶ D 4⟶L 5⟶ U 2⟶R 4⟶ U 1⟶L 7⟶ D 4⟶R 1⟶ D 5⟶L 2⟶ U 1⟶R 5⟶ D 2⟶L 1⟶ U 5⟶R 2⟶ U 7⟶L 4⟶ D 2⟶R 7⟶ U 4⟶L 2⟶ D 7⟶D 5⟶ R 1⟶U 4⟶ U 2⟶L 7⟶ D 5⟶R 2⟶ D 4⟶L 1⟶ U 2⟶U 5⟶ L 8.

Follow the steps above, I achieved the correct result.

Since the procedure seems to be a little messy... I didn't post it.

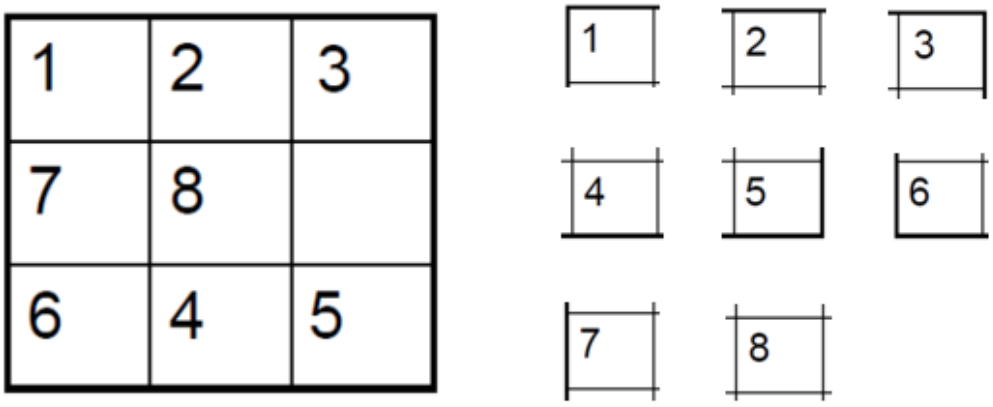


Figure 1: 8 Puzzle Result

3.2 BlocksWorld

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) )
    )
)
```

Result:

1. (move a f)
2. (move b e)
3. (move a c)
4. (move b a)
5. (move e f)
6. (move b d)
7. (move e b)
8. (move f a)

Apparently, the result above can reach the goal state.