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

E06 FF Planner

18340013 Conghao Chen

October 12, 2020

Contents

1 Examples

1.1 Spare Tire

domain_spare_tire.pddl

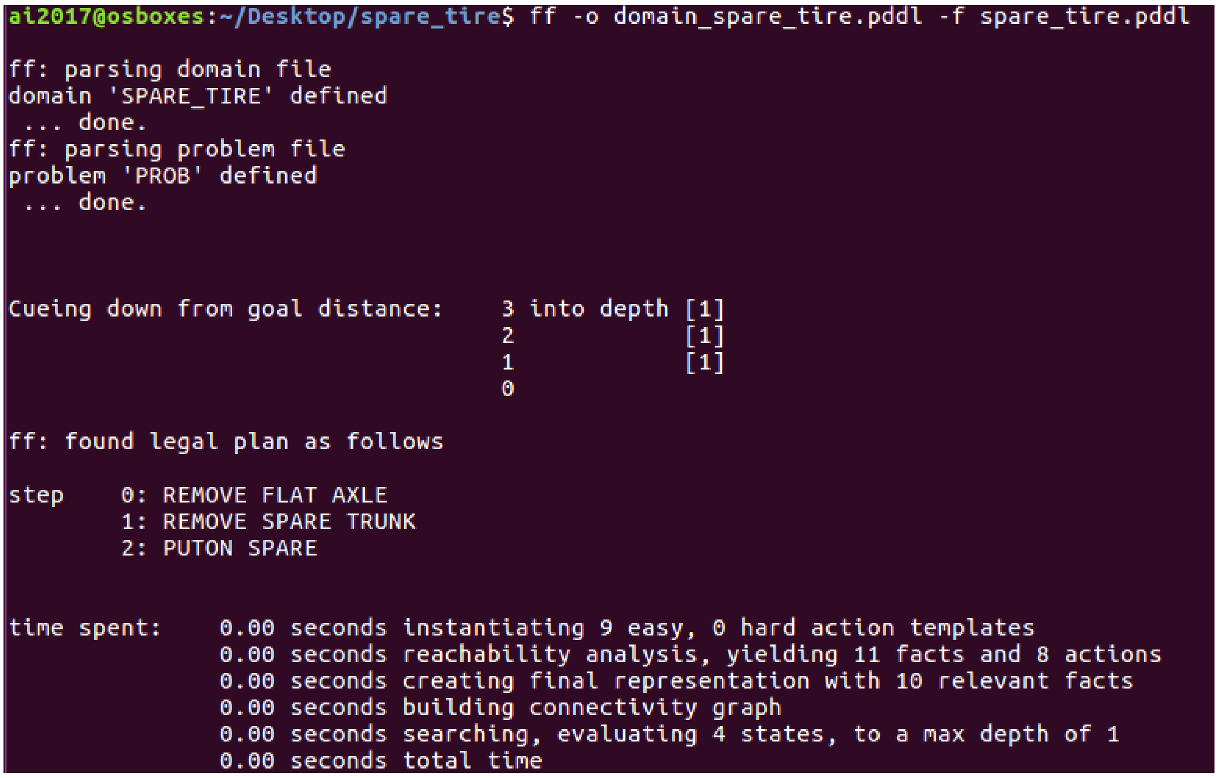
```
(define (domain spare_tire)
  (:requirements :strips :equality:typing)
  (:types physob location)
  (:predicates  (Tire ?x - physob)
                (at ?x - physob ?y - location))

(: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
```

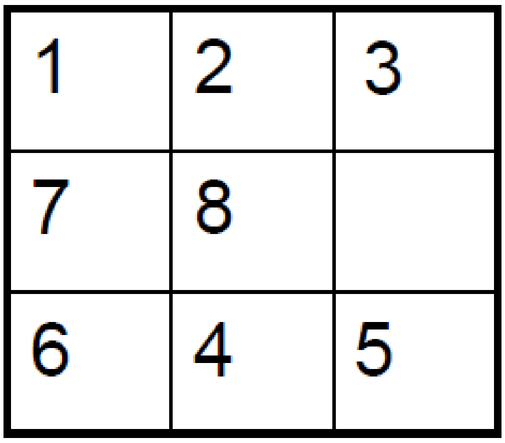
1.2 Briefcase World

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 ()
 )
)
```

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

3.1 8-puzzle

domain_puzzle.pddl

```
(define (domain puzzle)
(:requirements :strips :equality :typing)
(:types num loc)
(:predicates
(blank ?x)
(at ?n - num ?l - loc)
(neighbour ?l1 ?l2 - loc)
)

(:action slide ;
:parameters (?number - num ?from - loc ?to - loc)
:precondition (and
(at ?number ?from)
(blank ?to)
(or (neighbour ?from ?to) (neighbour ?to ?from))
)
:effect (and
(not (blank ?to))
(not (at ?number ?from))
(blank ?from)
(at ?number ?to)
)
)
)
```

puzzle.pddl

```
(define (problem prob)
(:domain puzzle)

(:objects
loc1 loc2 loc3 loc4 loc5 loc6 loc7 loc8 loc0 - loc
num1 num2 num3 num4 num5 num6 num7 num8 - num
)

(:init
; initial configuration
(at num1 loc1)
(at num2 loc2)
(at num3 loc3)
(at num7 loc4)
(at num8 loc5)
(at num6 loc7)
(at num4 loc8)
(at num5 loc0)
(blank loc6)

; define neighbourhoods
(neighbour loc1 loc2) (neighbour loc2 loc3)
(neighbour loc4 loc5) (neighbour loc5 loc6)
(neighbour loc7 loc8) (neighbour loc8 loc0)
(neighbour loc1 loc4) (neighbour loc4 loc7)
(neighbour loc2 loc5) (neighbour loc5 loc8)
(neighbour loc3 loc6) (neighbour loc6 loc0)
)

(:goal (and
(at num1 loc1) (at num2 loc2) (at num3 loc3)
(at num4 loc4) (at num5 loc5) (at num6 loc6)
(at num7 loc7) (at num8 loc8) (blank loc0)
))
)
```

3.2 Blocks World

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 (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))
    )))
))
```

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)) )
)
```

4 Results

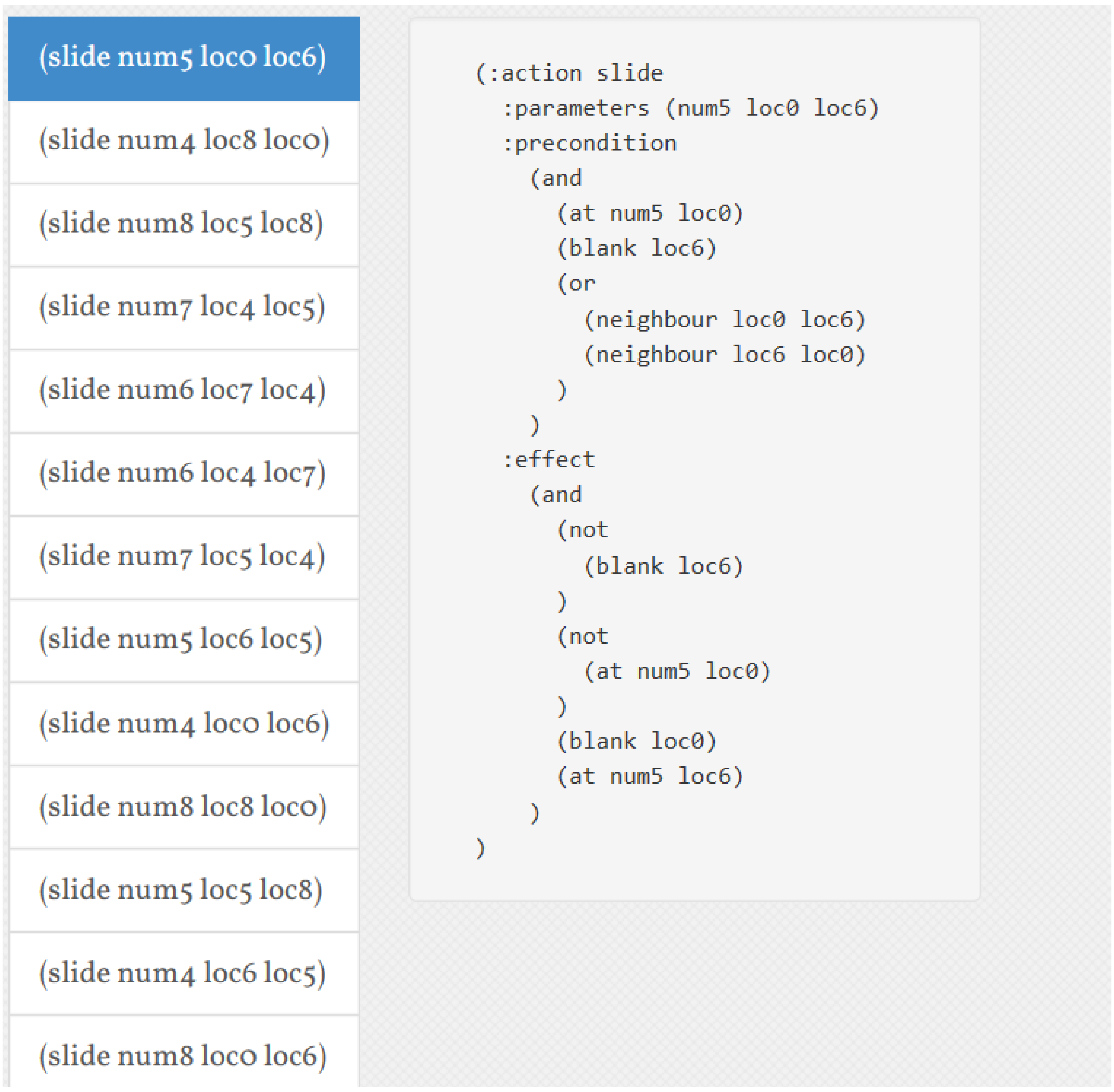
4.1 8-puzzle

在解决 `8-puzzle` 问题中, 定义了如下谓词:`blank(x)`, 表示 x 位置的块为空;`at(num,loc)` 表示数字 num 在块 loc 上;`neighbour(l1,l2)` 表示块 l1 与块 l2 相邻. 而 slide 的参数列表为 `slide(num,from,to)`, 表示 num 从块 from 移动到块 to; 前提和效果部分比较简单, 只要考虑全面即可, 详情见代码. 结果见 4.3:

4.2 Blocks World

`Blocks World` 的谓词和操作在前面已经给出, 而且文档提供的代码块已经实现了绝大部分, 只需要实现 `move` 和 `moveToTable` 即可. 主要考虑的问题就是一些参数既可以是桌子也可以是积木, 需要分情况讨论, 想明白这一点即可. 详情见代码. 结果见 4.3:

4.3 Appendix



(slide num5 loco loc6)

(slide num4 loc8 loco)

(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 loco loc6)

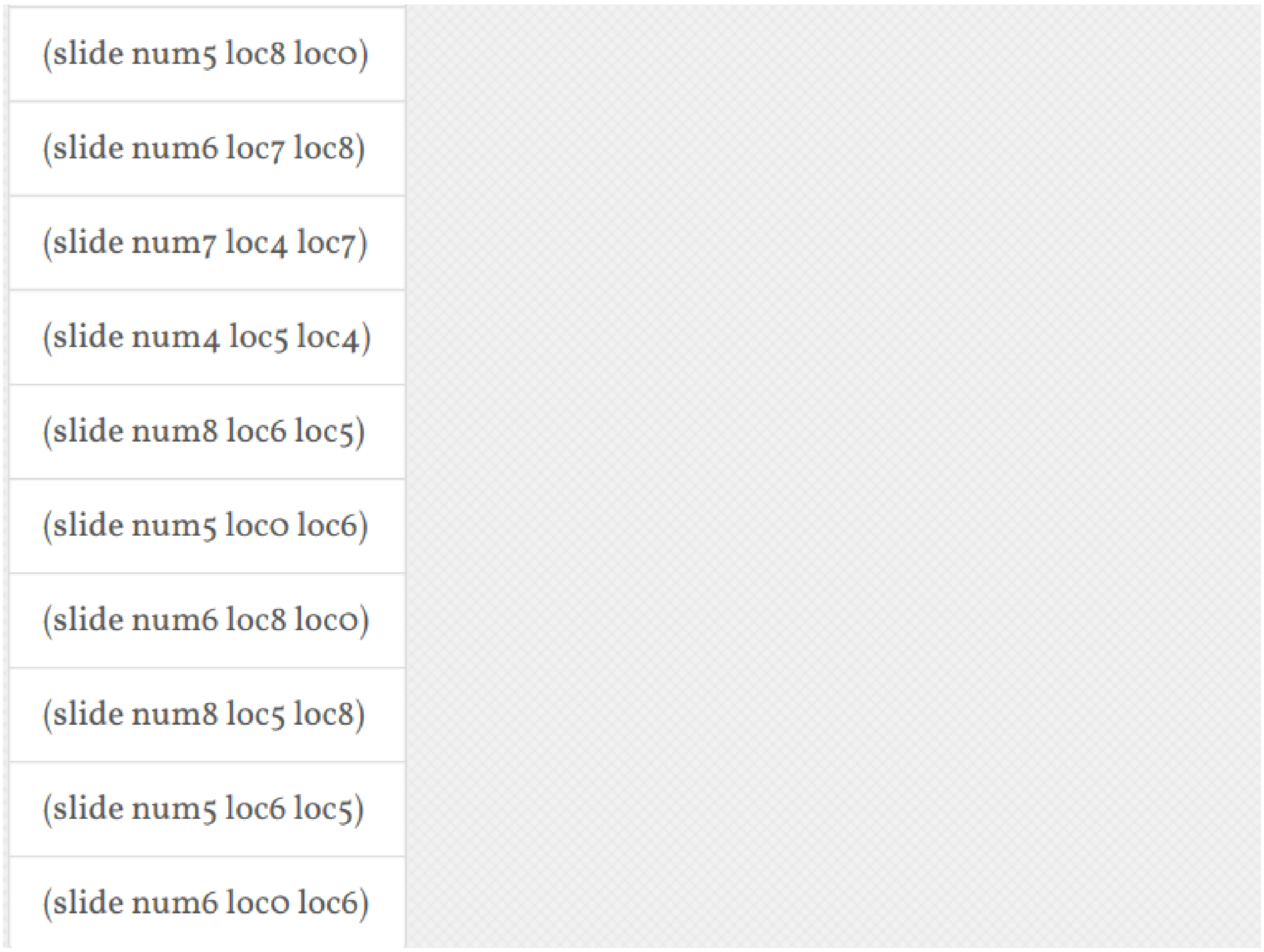
(slide num8 loc8 loco)

(slide num5 loc5 loc8)

(slide num4 loc6 loc5)

(slide num8 loco loc6)

```
(:action slide
  :parameters (num5 loc0 loc6)
  :precondition
    (and
      (at num5 loc0)
      (blank loc6)
      (or
        (neighbour loc0 loc6)
        (neighbour loc6 loc0)
      )
    )
  :effect
    (and
      (not
        (blank loc6)
      )
      (not
        (at num5 loc0)
      )
      (blank loc0)
      (at num5 loc6)
    )
)
```



(slide num5 loc8 loc0)

(slide num6 loc7 loc8)

(slide num7 loc4 loc7)

(slide num4 loc5 loc4)

(slide num8 loc6 loc5)

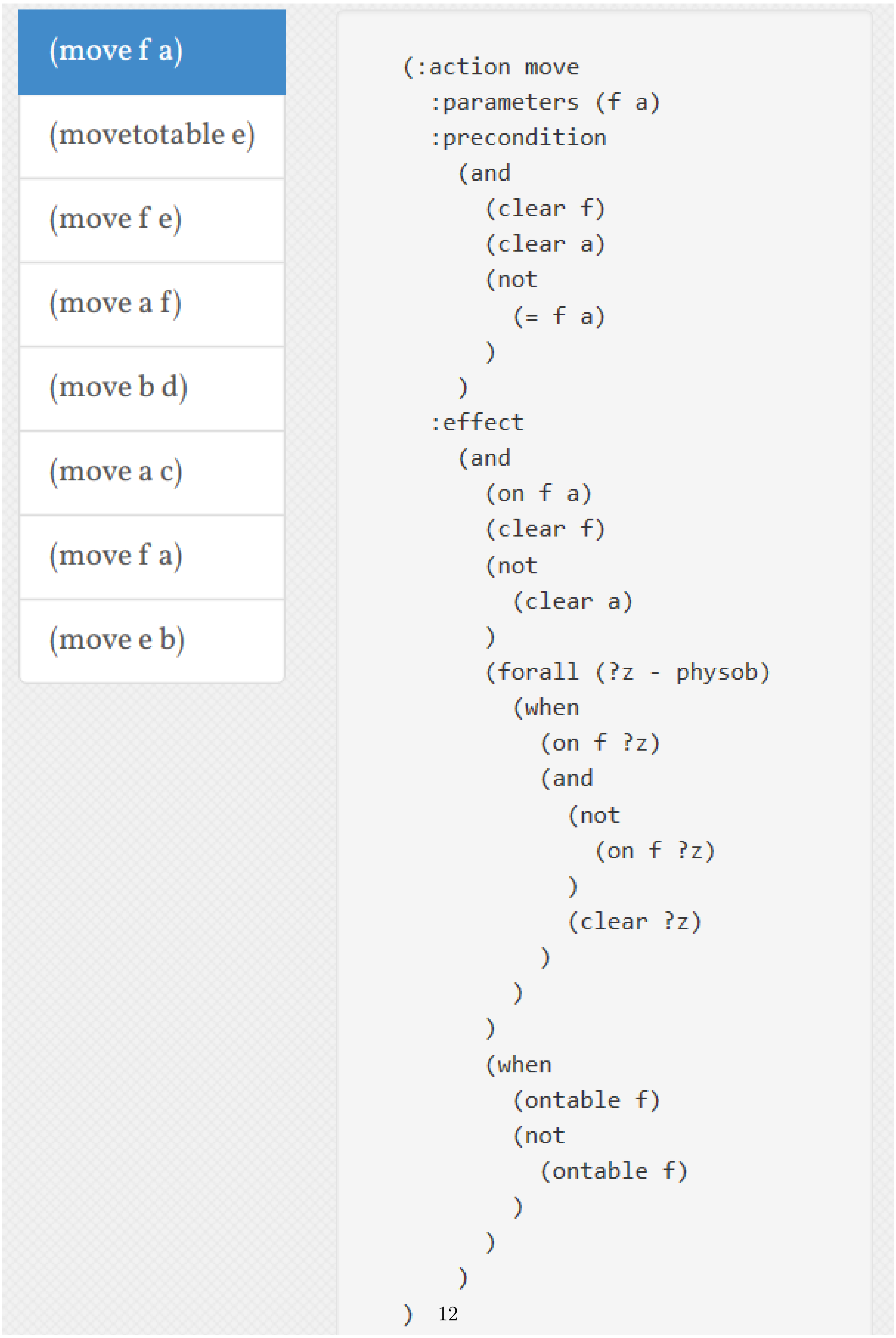
(slide num5 loc0 loc6)

(slide num6 loc8 loc0)

(slide num8 loc5 loc8)

(slide num5 loc6 loc5)

(slide num6 loc0 loc6)



(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)
        )
      )
    )
)
```