**Exercise C.1 (Monkey and Bananas)**

*A monkey is alone in a room, and the room has some bananas that hang from the ceiling. The monkey cannot reach the bananas, but in some far corner of the room there is also a chair that the monkey could use to reach them. The goal should be that the monkey has eaten the bananas.*

**DOMAIN FILE**

**Objects**

* Type *locatable*: monkey, monkey’s arms, bananas, chair
* Type *location*: position, height

**Predicates**

* at(x, y) – true iff *x* is *locatable*, *y* is a position and *x* is in *y*
* on(x, y) – true iff *x* is *locatable*, *y* is a height and *x* is in *y*
* is-eaten (x) – true iff *x* is a banana and *x* is grabbed by monkey
* free (x) – true iff *x* is monkey’s arm and *x* does not hold a banana
* grab (x, y) – true iff *x* is monkey, *y* a banana, and x grabs *y*

**PROBLEM FILE**

**Initial state**

* MONKEY(monkey) is true
* HAND(left), HAND(right), free(left) and free(right) are true
* BANANA(banana1), …, BANANA(banana4) are true
* CHAIR(chair) is true
* LOCATION(corner) and LOCATION(center) are true
* HEIGHT(down) and HEIGHT(high) are true
* at(monkey, center) and at(chair, corner) are true
* on(monkey, low) and on(chair, low) are true
* at(banana1, center), …, at(banana4, center) are true
* on(banana1, high), …, on(banana4, high) are true
* Everything else is false

**Goal specification**

* is-eaten (banana1), …, is-eaten(banana4) must be true
* Everything else we do not care about

**Actions/Operators**

* move(x, y)
  + *description*: monkey moves from position x to position y
  + *precondition:* at(monkey, x), on(monkey, floor)
  + *effect*: at(monkey, y), NOT at(monkey, x)
* push (x, y)
  + *description*: monkey pushes chair c from position x to position y
  + *precondition:* at(monkey, x), at(chair, x)
  + *effect*: at(monkey, y), at(chair, y), NOT at(monkey, x), NOT at(chair, x)
* climb (x, y)
  + *description*: monkey climb chair c from height x to height y
  + *precondition*: at(chair, center), at(monkey, center), on(monkey, x)
  + *effect*: on(monkey, y), NOT on(monkey, x)
* grab (x, y)
  + *description*:
  + *precondition*:
  + *effect*
* eat (x, y)
  + *description*:
  + *precondition*:
  + *effect*:

**CODE**

**Domain file**

(define (domain monkey)

(:requirements :strips :typing)

(:types

locatable location actuator - object

monkey banana chair - locatable

position height - location

hand - actuator

(:predicates

(at ?obj - locatable ?level - location)

(on ?obj - locatable ?height - level)

(is-eaten ?banana - locatable)

(free ?hand - actuator)

(grab ?monkey - locatable ?banana - locatable)

(:action

move

:parameters

(?x - position

?y - position)

:precondition

(and

(at monkey ?x)

(on monkey low)

)

:effect

(and

(at monkey ?y)

(not (at monkey ?x))

)

)

(:action

push

:parameters

(?x - position

?y - position)

:precondition

(and

(at chair ?x)

(at monkey ?x)

(on monkey low)

)

:effect

(

(at chair ?y)

(not (at chair ?x))

(at monkey ?y)

(not (at monkey ?x))

)

)

(:action

climb

:parameters

(?x - height  
 ?y - height)

:precondition

(and

(at chair center)

(at monkey center)

(on monkey ?x)

)

:effect

(and

(on monkey ?y)

(not (on monkey ?x))

)

)

(:action

grab

:parameters

(?x - hand

?y - banana)

:precondition

(and

(free ?x)

(on monkey high)

)

:effect

(and

(not (free ?x)

)

)

(:action

eat

:parameters

(?x - hand

?y - banana)

:precondition

(and

(not (free ?x))

(not (is-eaten ?y))

)

:effect

(

(free ?x)

(is-eaten ?y)

)

)

)

**Problem file**

(define (problem monkey-4bananas)

(:domain monkey)

(:objects

monkey - monkey

chair - chair

banana1 banana2 banana3 banana4 - banana

corner, center - position

low, high - height

left, right - hand

)

(:init

(on monkey, center) (at monkey, low)

(on chair, corner) (at chair, low)

(on banana1 center) (at banana1, high)

(on banana2 center) (at banana2, high)

(on banana3 center) (at banana3, high)

(on banana4 center) (at banana4, high)

(free right) (free left)

)

(:goal

(is-eaten banana1)

(is-eaten banana2)

(is-eaten banana3)

(is-eaten banana4)

)

)

**Exercise C.2 (Teleporting Sokoban)**

*The player is confined to the board and may move horizontally or vertically onto empty squares (never through walls or boxes). The player can also move into a box, which pushes it into the square beyond. Boxes may not be pushed into other boxes or walls, and they cannot be pulled. The number of boxes is equal to the number of storage locations. The puzzle is solved when all boxes are at storage locations.*

**DOMAIN FILE**

**Objects**

* Type *locatable*: agent, box
* Type *location*: goal square, normal square,

**Predicates**

* at(x, y) – true iff *x* is *locatable*, *y* is a position and *x* is in *y*
* on(x, y) – true iff *x* is *locatable*, *y* is a height and *x* is in *y*
* is-goal (x) – true iff *x* is a banana and *x* is grabbed by monkey
* free (x) – true iff *x* is monkey’s arm and *x* does not hold a banana
* grab (x, y) – true iff *x* is monkey, *y* a banana, and x grabs *y*

**PROBLEM FILE**

**Initial state**

**Goal specification**

* All boxes in goal squares

**Actions/operators**

* Teleport