**Robot**

**Domain**

(define (domain gripper-strips)

(:predicates

(room ?r)

(ball ?b)

(gripper ?g)

(at-robby ?r)

(at ?b ?r)

(free ?g)

(carry ?o ?g)

)

(:action move

:parameters (?from ?to)

:precondition (and (room ?from)(room ?to)(at-robby ?from))

:effect (and (at-robby ?to)

(not(at-robby ?from)))

)

(:action pick

:parameters (?obj ?room ?gripper)

:precondition (and (ball ?obj)(room ?room)

(gripper ?gripper)

(at ?obj ?room)

(at-robby ?room)(free ?gripper))

:effect (and (carry ?obj ?gripper)

(not (at ?obj ?room))

(not (free ?gripper)))

)

(:action drop

:parameters (?obj ?room ?gripper)

:precondition (and (ball ?obj)(room ?room)(gripper ?gripper)

(carry ?obj ?gripper)(at-robby ?room))

:effect (and (at ?obj ?room)(free ?gripper)

(not (carry ?obj ?gripper)))

)

)

**Problem**

(define (problem strips-gripper2)

(:domain gripper-strips)

(:objects

rooma roomb ball1 ball2 left right

)

(:init

(room rooma)(room roomb)

(ball ball1)(ball ball2)

(gripper left)(gripper right)

(at-robby rooma)(free left)(free right)

(at ball1 rooma)

(at ball2 rooma)

)

(:goal

(and (at ball1 roomb))

)

)

**Problem 2**

(define(problem strips-gripper4)

(:domain gripper-strips)

(:objects

rooma roomb ball1 ball2 ball3 ball4 right left

)

(:init

(room rooma)(room roomb)

(ball ball1)(ball ball2)(ball ball3)(ball ball4)

(gripper right)(gripper left)(at-robby rooma)

(free right)(free left)

(at ball1 rooma)

(at ball2 rooma)

(at ball3 rooma)

(at ball4 rooma)

)

(:goal

(and(at ball1 roomb)

(at ball2 roomb)

(at ball3 roomb))

)

)

**Problem 3**

(define(problem strips-gripper5)

(:domain gripper-strips)

(:objects

rooma roomb ball1 ball2 ball3 ball4 ball5 ball6 right left

)

(:init

(room rooma)(room roomb)(ball ball1)(ball ball2)(ball ball3)(ball ball4)(ball ball5)(ball ball6)

(gripper left)(gripper right)(at-robby rooma)

(free left)(free right)

(at ball1 rooma)

(at ball2 rooma)

(at ball3 rooma)

(at ball4 rooma)

(at ball5 rooma)

(at ball6 rooma)

)

(:goal

(and

(at ball1 roomb)

(at ball2 roomb)

(at ball3 roomb)

(at ball4 roomb)

(at ball5 roomb)

)

)

)

**Problem 4**

(define(problem strips-gripper8)

(:domain gripper-strips)

(:objects

rooma roomb ball1 ball2 ball3 ball4 ball5 ball6 ball7 ball8 right left

)

(:init

(room rooma)(room roomb)(ball ball1)(ball ball2)(ball ball3)(ball ball4)(ball ball5)(ball ball6)(ball ball7)(ball ball8)

(gripper left)(gripper right)(at-robby rooma)(free left)(free right)

(at ball1 rooma)

(at ball2 rooma)

(at ball3 rooma)

(at ball4 rooma)

(at ball5 rooma)

(at ball6 rooma)

(at ball7 rooma)

(at ball8 rooma)

)

(:goal

(and(at ball1 roomb)

(at ball2 roomb)

(at ball3 roomb)

(at ball4 roomb)

(at ball5 roomb)

(at ball6 roomb)

(at ball7 roomb))

)

)

**Puzzel**

**Domain**

(define (domain strips-sliding-tile)

(:requirements :strips)

(:predicates

(tile ?x)(position ?x)

(at ?t ?x ?y)(blank ?x ?y)

(inc ?p ?pp)(dec ?p ?pp)

)

(:action move-up

:parameters (?ex ?ey ?t ?tx)

:precondition (and

(tile ?t)(position ?ex)(position ?ey)(position ?tx)

(dec ?ex ?tx)(blank ?ex ?ey) (at ?t ?tx ?ey)

)

:effect (and

(not(blank ?ex ?ey)) (not(at ?t ?tx ?ey))

(blank ?tx ?ey)(at ?t ?ex ?ey)

)

)

(:action move-down

:parameters (?ex ?ey ?t ?tx)

:precondition (and

(tile ?t)(position ?ex)(position ?ey)(position ?tx)

(inc ?ex ?tx)(blank ?ex ?ey)(at ?t ?tx ?ey)

)

:effect (and

(not(blank ?ex ?ey))(not(at ?t ?tx ?ey))

(blank ?tx ?ey)(at ?t ?ex ?ey)

)

)

(:action move-left

:parameters (?ex ?ey ?t ?ty)

:precondition (and

(tile ?t)(position ?ex)(position ?ey)(position ?ty)

(dec ?ey ?ty)(blank ?ex ?ey)(at ?t ?ex ?ty)

)

:effect (and

(not(blank ?ex ?ey))(not(at ?t ?ex ?ty))

(blank ?ex ?ty)(at ?t ?ex ?ey)

)

)

(:action move-right

:parameters (?ex ?ey ?t ?ty)

:precondition (and

(tile ?t)(position ?ex)(position ?ey)(position ?ty)

(inc ?ey ?ty)(blank ?ex ?ey)(at ?t ?ex ?ty)

)

:effect (and

(not(blank ?ex ?ey))(not(at ?t ?ex ?ty))

(blank ?ex ?ty)(at ?t ?ex ?ey)

)

)

)

**Problem**

(define (problem hard1)

(:domain strips-sliding-tile)

(:objects

t1 t2 t3 t4 t5 t6 t7 t8 x1 x2 x3 y1 y2 y3

)

(:init

(tile t1)

(tile t2)

(tile t3)

(tile t4)

(tile t5)

(tile t6)

(tile t7)

(tile t8)

(position x1)

(position x2)

(position x3)

(position y1)

(position y2)

(position y3)

(inc x1 x2)

(inc x2 x3)

(dec x3 x2)

(dec x2 x1)

(inc y1 y2)

(inc y2 y3)

(dec y3 y2)

(dec y2 y1)

(blank x1 y2)

(at t8 x1 y1)

(at t7 x2 y1)

(at t6 x3 y1)

(at t4 x2 y2)

(at t1 x3 y2)

(at t2 x1 y3)

(at t5 x2 y3)

(at t3 x3 y3)

)

(:goal (and

(at t1 x2 y1)

(at t2 x3 y1)

(at t3 x1 y2)

(at t4 x2 y2)

(at t5 x3 y2)

(at t6 x1 y3)

(at t7 x2 y3)

(at t8 x3 y3)

)

)

)

**HaNoi Tower**

**Domain**

(define (domain hanoi)

(:requirements :strips)

(:predicates

(clear ?x)

(on ?x ?y)

(larger ?x ?y)

)

(:action move

:parameters (?disc ?from ?to)

:precondition (and

(larger ?to ?disc)

(on ?disc ?from)

(clear ?disc)

(clear ?to)

)

:effect (and

(clear ?from)

(on ?disc ?to)

(not (on ?disc ?from))

(not (clear ?to))

)

)

)

**Problem 1**

(define (problem hanoi3)

(:domain hanoi)

(:objects

peg1 peg2 peg3 d1 d2 d3

)

(:init

(larger peg1 d1)

(larger peg1 d2)

(larger peg1 d3)

(larger peg2 d1)

(larger peg2 d2)

(larger peg2 d3)

(larger peg3 d1)

(larger peg3 d2)

(larger peg3 d3)

(larger d3 d2)

(larger d3 d1)

(larger d2 d1)

(clear peg2)

(clear peg3)

(clear d1)

(on d3 peg1)

(on d2 d3)

(on d1 d2)

)

(:goal

(and(on d3 peg3) (on d2 d3) (on d1 d2))

)

)

**Problem 2**

(define (problem hanoi4)

(:domain hanoi)

(:objects

peg1 peg2 peg3 d1 d2 d3 d4

)

(:init

(larger peg1 d1)

(larger peg1 d2)

(larger peg1 d3)

(larger peg1 d4)

(larger peg2 d1)

(larger peg2 d2)

(larger peg2 d3)

(larger peg2 d4)

(larger peg3 d1)

(larger peg3 d2)

(larger peg3 d3)

(larger peg3 d4)

(larger d4 d3)

(larger d4 d2)

(larger d4 d1)

(larger d3 d2)

(larger d3 d1)

(larger d2 d1)

(clear peg2)

(clear peg3)

(clear d1)

(on d4 peg1)

(on d3 d4)

(on d2 d3)

(on d1 d2)

)

(:goal

(and(on d4 peg3) (on d3 d4) (on d2 d3) (on d1 d2))

)

)

**Air Cargo Transport**

**Domain**

(define (domain air-cargo)

(:requirements :strips)

(:predicates

(In ?obj ?place)

(At ?obj ?place)

(Cargo ?obj)

(Plane ?obj)

(Airport ?obj)

)

(:action LOAD

:parameters (?c ?p ?a)

:precondition (and

(At ?c ?a) (At ?p ?a)

(Cargo ?c) (Plane ?p) (Airport ?a)

)

:effect (and

(not (At ?c ?a)) (In ?c ?p)

)

)

(:action UNLOAD

:parameters (?c ?p ?a)

:precondition (and

(In ?c ?p) (At ?p ?a)

(Cargo ?c) (Plane ?p) (Airport ?a)

)

:effect (and

(not (In ?c ?p)) (At ?c ?a)

)

)

(:action FLY

:parameters (?p ?from ?to)

:precondition (and

(At ?p ?from) (Plane ?p) (Airport ?from) (Airport ?to)

)

:effect (and

(not (At ?p ?from)) (At ?p ?to)

)

)

)

**Problem**

(define (problem pl1)

(:domain air-cargo)

(:objects

c1 c2 P1 P2 SFO JFK

)

(:init

;;type

(Cargo c1)

(Cargo c2)

(Plane P1)

(Plane P2)

(Airport SFO)

(Airport JFK)

;;location

(At c1 SFO)

(At c2 JFK)

(At P1 SFO)

(At P2 JFK)

)

(:goal

(and (At c1 JFK) (At c2 SFO))

)

)