Planning using PPDL

Artificial Intelligence and Knowledge Engineering – List No. 3

The purpose of the list

The purpose of the list is to get familiarized with the language **PDDL** (Planning Domain Definition Language).

The list consists of three simple tasks.

Theoretical introduction

The Planning Domain Definition Language (PDDL) is a formal language for describing planning problems used in artificial intelligence, especially in the field of symbolic planning. PDDL uses a LISP-inspired prefix notation (S-expressions).

The PDDL definition consists of two parts:

1. domain definition (domain.pddl),
2. problem definition (problem.pddl).

Sample `domain.pddl`

```
(define (domain robot-world)
  (:requirements :strips :typing)
  (:types location robot)

  (:predicates
    (at ?r - robot ?l - location)
    (connected ?from ?to - location)
  )

  (:action move
    :parameters (?r - robot ?from ?to - location)
    :precondition (and (at ?r ?from) (connected ?from ?to))
    :effect (and (not (at ?r ?from)) (at ?r ?to))
  )
)
```

Sample `problem.pddl`

```
(define (problem robot-move-problem)
```

```
  (:domain robot-world)

  (:objects
    room1 room2 room3 - location
    robby - robot
  )

  (:init
    (at robby room1)
    (connected room1 room2)
    (connected room2 room3)
    (connected room3 room1)
  )

  (:goal (at robby room3))
)
```

Although not required by the PDDL standard, many planners require these two parts to be in separate files.

PDDL (Planning Domain Definition Language) is a domain description language used in Artificial Intelligence Planning. It allows you to:

- defining the domain (available actions and their effects),
- describing the problem (initial state and goal),
- automatically generating an action plan by planners (np. Fast Downward, pyperplan).

How to run PDDL?

You can use a planner like Fast Downward or an online editor:

- `fast-downward.py domain.pddl problem.pddl - search "astar(lmcut)"`
- Online editor: `https://editor.planning.domains`

Task 1 (max. 15 points):

The task is to design, run, and analyze a package transport plan using the **PDDL** language with extensions. The goal is to develop a logical model and implement it in the form of PDDL files, and then test the planner's operation and evaluate the generated plan. The following PDDL extensions are used in the project:

- `:strips` – basic STRIPS operation model,
- `:typing` – object typing (e.g. `package`, `location`, `vehicle`),
- `:negative-preconditions` – negative conditions (e.g. `(not (at package location))`),
- `:conditional-effects` – optional, conditional effects,

- `:multi-agent` – for an extended model with multiple vehicles.
- action costs (`:numeric-fluents`, `:action-costs`),
- duration (`:durative-actions`),
- complex transport topology (road, air and water).

Task 2 (max. 10 points):

A cleaning robot that has to visit all the rooms and vacuum them.

- **Objects:** robot `robo`, rooms: `room1`, `room2`, `room3`.
- **Predicates:**
  - `(at ?r - robot ?p - room)` – robot is in the room;
  - `(dirty ?p - room)` – room is dirty;
  - `(clean ?p - room)` – room is clean.
- **Actions:**
  - `move` – moves the robot between rooms;
  - `clean` – cleans the current room.
- **Goal:** All rooms are clean: `(and (clean room1) (clean room2) (clean room3))`

Task 3 (max. 10 pkt.):

Based on the lecture and materials from this page solve the following task:

Task description:

The robot has two arms and can move between two rooms. The first room contains four balls and the robot itself. The goal is to move all the balls to the second room using the actions of picking up and putting down the balls.

Objects:

- **Rooms:** `room1`, `room2`
- **Balls:** `ball1`, `ball2`, `ball3`, `ball4`
- **Robot arms:** `arm1`, `arm2`
- **Robot:** `robot`

Predicates:

- `(at ?r - robot ?room - room)` – robot in the room;
- `(inroom ?b - ball ?room - room)` – ball in the room;
- `(holding ?a - arm ?b - ball)` – arm holds the ball;
- `(arm-empty ?a - arm)` – arm is empty.

File `domain.pddl`:

```
(define (domain ball-moving-robot)
  (:requirements :strips :typing)
  (:types robot room ball arm)

  (:predicates
    (at ?r - robot ?rm - room)
    (inroom ?b - ball ?rm - room)
    (holding ?a - arm ?b - ball)
    (arm-empty ?a - arm)
  )

  (:action move
    :parameters (?r - robot ?from - room ?to - room)
    :precondition (at ?r ?from)
    :effect (and (not (at ?r ?from)) (at ?r ?to))
  )

  (:action pick-up
    :parameters (?r - robot ?a - arm ?b - ball ?rm - room)
    :precondition (and (at ?r ?rm) (inroom ?b ?rm) (arm-empty ?a))
    :effect (and (holding ?a ?b) (not (arm-empty ?a)) (not (inroom
      ↪ ?b ?rm)))
  )

  (:action put-down
    :parameters (?r - robot ?a - arm ?b - ball ?rm - room)
    :precondition (and (at ?r ?rm) (holding ?a ?b))
    :effect (and (inroom ?b ?rm) (arm-empty ?a) (not (holding ?a ?b
      ↪ )))
  )
)
```

File `problem.pddl`:

```
(define (problem move-balls)
  (:domain ball-moving-robot)

  (:objects
    room1 room2 - room
```

```
    robot - robot
    ball1 ball2 ball3 ball4 - ball
    arm1 arm2 - arm
  )

  (:init
    (at robot room1)
    (inroom ball1 room1)
    (inroom ball2 room1)
    (inroom ball3 room1)
    (inroom ball4 room1)
    (arm-empty arm1)
    (arm-empty arm2)
  )

  (:goal
    (and
      (inroom ball1 room2)
      (inroom ball2 room2)
      (inroom ball3 room2)
      (inroom ball4 room2)
    )
  )
)
```

What needs to be uploaded?

- PDF report with problem description, plan screenshot, experiment analysis and conclusions;
- Modified files `.pddl` .

Deadline and form of submission:

- Deadline: *In the 10th labs.*
- Form: via Moodle.

Additional materials:

- http://editor.planning.domains
- https://www.fast-downward.org/
- https://planning.wiki/ref/pddl/domain
- https://fareskalaboud.github.io/LearnPDDL/
- https://www.cs.toronto.edu/~sheila/2542/s14/A1/introtopddl2.pdf

- http://pddl4j.imag.fr/pddl_tutorial.html