Planowanie z wykorzystaniem języka PPDL

Sztuczna Inteligencja i Inżynieria Wiedzy – Lista nr 3

Cel listy

Celem listy jest jest zapoznanie się z językiem **PDDL** (Planning Domain Definition Language).

Lista składa się z trzech prostych zadań.

Wprowadzenie teoretyczne

Język PDDL (Planning Domain Definition Language) to formalny język opisu problemów planowania używany w sztucznej inteligencji, zwłaszcza w dziedzinie planowania symbolicznego. PDDL korzysta z notacji prefiksowej inspirowanej językiem LISP (tzw. S-wyrażenia).

Definicja PDDL składa się z dwóch części:

1. definicji domeny (domain.pddl),
2. definicji problemu (problem.pddl).

Przykład `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))
  )
)
```

Przykład `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))
)
```

Chociaż nie jest to wymagane przez standard PDDL, wielu planistów wymaga, aby te dwie części znajdowały się w oddzielnych plikach.

PDDL (Planning Domain Definition Language) to język opisu dziedzin używany w automatycznym planowaniu (Artificial Intelligence Planning). Pozwala na:

- zdefiniowanie dziedziny (dostępnych akcji i ich efektów),
- opisanie problemu (stan początkowy i cel),
- automatyczne generowanie planu działania przez planery (np. Fast Downward, pyperplan).

Jak uruchomić PDDL?

Można skorzystać z planera np. Fast Downward lub edytora online:

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

Zadanie 1 (max. 15 pkt.):

Zadanie polega na zaprojektowaniu, uruchomienie i analizie planu transportu paczek z wykorzystanierm języka **PDDL** z rozszerzeniami. Celem jest opracowanie modelu logicznego oraz wdrożenie go w formie plików PDDL, a następnie przetestowanie działania planera i ocena wygenerowanego planu. W projekcie wykorzystywane są następujące rozszerzenia języka PDDL:

- `:strips` – podstawowy model operacji STRIPS,
- `:typing` – typowanie obiektów (np. `paczka`, `lokacja`, `pojazd`),

- `:negative-preconditions` – warunki negatywne (np. `(not (at paczka miejsce))`),
- `:conditional-effects` – opcjonalnie, efekty warunkowe,
- `:multi-agent` – przy modelu rozszerzonym z wieloma pojazdami.
- kosztów działań (`:numeric-fluents`, `:action-costs`),
- czasu trwania (`:durative-actions`),
- złożonej topologii transportu (drogowego, lotniczego oraz wodnego).

Zadanie 2 (max. 10 pkt.):

Sprzątający robot, który ma odwiedzić wszystkie pokoje i odkurzyć je.

- **Obiekty:** robot `robo`, pokoje: `pokoj1`, `pokoj2`, `pokoj3`.
- **Predykaty:**
  - `(at ?r - robot ?p - room)` – robot znajduje się w pokoju
  - `(dirty ?p - room)` – pokój jest brudny
  - `(clean ?p - room)` – pokój jest czysty.
- **Akcje:**
  - `move` – przemieszcza robota między pokojami;
  - `clean` – sprząta aktualny pokój.
- **Cel:** Wszystkie pokoje są czyste: `(and (clean pokoj1) (clean pokoj2) (clean pokoj3))`

Zadanie 3 (max. 10 pkt.):

Na podstawie wykładu oraz materiałow z tej strony rozwiąż następujące zadanie:

Opis zadania:

Robot posiada dwa ramiona i może poruszać się między dwoma pokojami. W pierwszym pokoju znajdują się cztery piłki oraz sam robot. Celem jest przeniesienie wszystkich piłek do drugiego pokoju, wykorzystując akcje podnoszenia i odkładania piłek.

Obiekty:

- **Pokoje:** `room1`, `room2`
- **Piłki:** `ball1`, `ball2`, `ball3`, `ball4`
- **Ramiona robota:** `arm1`, `arm2`
- **Robot:** `robot`

Predykaty:

- `(at ?r - robot ?room - room)` – robot w pokoju;
- `(inroom ?b - ball ?room - room)` – piłka w pokoju;
- `(holding ?a - arm ?b - ball)` – ramię trzyma piłkę;
- `(arm-empty ?a - arm)` – ramię jest puste.

Plik `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
      ↪ )))
  )
)
```

Plik `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)
    )
  )
)
```

Co należy oddać?

- Raport PDF z opisem problemu, zrzutem planu, analizą eksperymentów i wnioskami;
- Zmodyfikowane pliki `.pddl` .

Termin i forma oddania:

- Termin: *Na 10. zajęciach.*
- Forma: przez Moodle.

Materiały dodatkowe:

- 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