Künstliche Intelligenz
– Aufgabenblatt 06 –

Prof. Dr. David Spieler
Hochschule München

9. März 2023

In diesem Aufgabenblatt werden wir einige Planungsprobleme mit dem STRIPS Planer Pyperplan lösen. Dieser Planer wurde in Python geschrieben und Sie können ihn in der Kommandozeile mit Hilfe von `pip install pyperplan` installieren. Die Aufgabenbeschreibung erfolgt in der eigenen Sprache PDDL, der Planning Domain Definition Language, dem Standard für klassische Planungsprobleme und zwar in Form von einer gemeinsamen Domain Datei und je einer eigenen Problem Datei.

Eine Domain Datei beinhaltet die Spezifikation der Prädikate und der Aktionen nach dem Muster

```
(define (domain <domain name>)
  <predicates>
  <first action>
  ...
  <last action>
)
```

wobei `<domain name>` eine Zeichenkette ist, welcher der Planungsdomäne eine Namen gibt. Eine Problem Datei beinhaltet Objekte, den Startzustand und das Ziel nach dem Muster

```
(define (problem <problem name>)
  (:domain <domain name>)
  <objects>
  <initial state>
  <goal specification>
)
```

wobei `<problem name>` dem Problem einen Namen gibt und `<domain name>` sich auf eine Planungsdomäne in einer Domain Datei bezieht.

Wir betrachten nun ein Standardbeispiel – die Gripper Domäne und ein passendes Problem dazu. Es handelt sich dabei um einen Agenten *Robby*, der Greifer (*gripper*)

besitzt, mit denen er Bälle aufnehmen und ablegen kann. Zudem kann er sich zwischen Räumen hin und her bewegen. Die Domain Datei `gripper.pddl` beinhaltet

```
(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)))
    )
)
```

Zunächst werden mit `(:predicates ... )` die Prädikate spezifiziert. Dazu gehören beispielsweise `(room ?r)` – `?r` ist ein Raum, `(at-robby ?r)` – der Agent befindet sich in Raum `?r` oder `(at ?b ?r)` – Ball `?b` befindet sich in Raum `?r`. Danach folgen die Aktionsspezifikationen nach dem Schema

```
(:action <name>
    :parameters (...)
    :precondition (...)
    :effect (...)
)
```

Die Aktion `move` beispielsweise kodiert die Bewegung des Agenten von einem Raum `?from` (Parameter) in den Raum `?to` (Parameter). Damit die Aktion durchgeführt werden kann, muss die Voraussetzung (`precondition`) erfüllt sein, dass `?from` ein Raum ist – `(room ?from)`, `?to` ein Raum ist – `(room ?to)` und (`and`) der Agent sich in Raum `?from` befindet – `(at-robby ?from)`. Der Effekt `effect` ist, dass sich der Agent danach in Raum `?to` befindet – `(at-robby ?to)` und (`and`) nicht mehr in Raum `?from` – `(not (at-robby ?from))`.

Die Problem Datei `gripper-two.pddl` beinhaltet

```
(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))
       )
)
```

Zunächst werden in `(:objects ...)` alle Objekte als Namen aufgelistet. Erst in der Spezifikation der Anfangsbedigungen in `(:init ...)` werden deren Bedeutungen mit Hilfe der Prädikate aus der Domänen Datei festgelegt. Beispielsweise gibt es zwei Räume `rooma` und `roomb` mit `(room rooma)` und `(room roomb)`. Zudem werden die Aufenthaltsorte festgelegt, so befinden sich der Agent und beide Bälle anfangs in `rauma` wegen `(at-robby rooma)`, `(at ball1 rooma)` und `(at ball2 roomb)`. Das Ziel spezifiziert in

`(:goal ...)` ist, dass sich `ball1` in `roomb` befindet.

Wenn Sie mit Hilfe von `pyperplan gripper.pddl gripper-two.pddl` eine Lösung anfordern, erhalten Sie die Lösungsdatei `gripper-two.pddl.soln` mit der Lösungssequenz

```
(pick ball1 rooma left)
(move rooma roomb)
(drop ball1 roomb left)
```

Das bedeutet, der Agent nimmt via `pick` zunächst `ball1` in `rooma` mit dem linken Greifer `left` auf (Zeile 1). Danach bewegt er sich von `rooma` zu `roomb` (Zeile 2). Dort legt `drop` er `ball1` aus dem linken `left` Greifer in `roomb` ab (Zeile 3).

Aufgabe 1 (Gripper) *Installieren Sie zunächst `pyperplan`.*

1. *Führen Sie nun das `Gripper-Two` Beispiel aus und überprüfen, dass bei Ihnen `pyperplan` auch eine Lösung findet. Lassen Sie das Programm mehrmals laufen - ist es immer die gleiche Lösung?*
2. *Ändern Sie das Problem, sodass nun beide Bälle in `roomb` sein müssen. Lassen Sie einen Plan erstellen.*
3. *Ändern Sie das Problem, sodass der Agent nur noch einen Greifer `middle` hat, die beiden Bälle anfangs in `rooma` sind und am Ende in `roomb`. Lassen Sie einen Plan erstellen.*

Aufgabe 2 (Boxes) *Schreiben Sie eine Domain Datei `boxel.pddl` mit*

- *den Prädikatdefinitionen `box ?b`, `free ?o` und `on ?o1 ?o2`, die jeweils festlegen, dass `?b` eine Box ist, Objekt `?o` frei ist (also kein Objekt oben darauf sitzt) bzw. Objekt `?o1` sich auf Objekt `?o2` befindet.*
- *einer Aktion `move` mit den Parametern `?box`, `?from` und `?to`, welche die Box `?box` von `?from` nach `?to` bewegt. Entwickeln Sie hierfür sinnvolle Voraussetzungen und Effekte.*

Schreiben Sie eine Problem Datei `boxes-problem.pddl`, in der Sie drei Positionen `left`, `middle` und `right` und drei Boxen `boxa`, `boxb` und `boxc` definieren. Anfangs befindet sich `boxc` auf `boxb` auf `boxa` auf Position `left`.

1. *Lassen Sie sich einen Plan für das Ziel generieren, bei dem `boxa` auf `boxb` auf `boxc` auf der rechten Position befindet.*
2. *Lassen Sie sich einen Plan für das Ziel generieren, bei dem `boxa` auf `boxc` auf `boxb` auf der rechten Position befindet.*