Bachelor Projekt — Grundlagen der Künstlichen Intelligenz

B. Nebel, T. Engesser, R. Mattmüller, D. Speck
Universität Freiburg
Wintersemester 2019/20
Institut für Informatik

Übungsblatt 3

Aufgabe 3.1 (Automatische Handlungsplanung)

Suchprobleme in strukturierten Zustandsräumen lassen sich häufig als Handlungsplanungsaufgaben darstellen: Gegeben eine Beschreibung des Ist-Zustandes der Welt, des Soll-Zustandes der Welt, und der verfügbaren Aktionen, um den Zustand der Welt zu verändern, finde einen Plan (= eine Folge von Aktionen), der den Ist-Zustand in den Soll-Zustand überführt.
Verwendet man dazu nun ein Handlungsplanungs-Programm wie etwa FAST DOWNWARD[1], so muss man sich nicht mehr um die Suche selbst kümmern. Vielmehr muss man nur noch die verfügbaren Aktionen, den Ist- und den Soll-Zustand in einer geeigneten Eingabesprache (in der Regel PDDL, die *Planning Domain Definition Language*) deklarativ aufschreiben und kann den Rest dem Planer überlassen.
Wir wollen diesen Weg anhand eines konkreten Beispiels gehen: KAMI[2] bzw. KAMI 2[3].

(a) Machen Sie sich mit PDDL vertraut. Einige Dokumente, in denen die Sprache beschrieben wird, finden Sie auf der Website des Internationalen Planungswettbewerbs 2008[4].

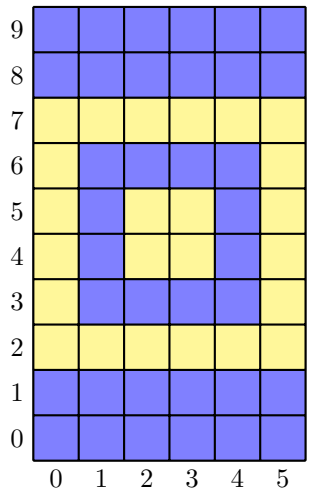
Eine sehr gute Spielwiese zum Erlernen von PDDL ist der Web-basierte PDDL-Editor[5]. Dort können Sie eigene PDDL-Dateien erstellen oder bestehende von vergangenen Planungswettbewerben über *Import* importieren, modifizieren usw. Importieren Sie beispielsweise *IPC-1998* → *gripper* → *prob01.pddl* → *Import both* und versuchen Sie, die Problembeschreibung zu verstehen. Mittels *Solve* können Sie Ihre erstellten oder importierten Planungprobleme mit einem eingebauten Planer lösen lassen. Beachten Sie, dass der dort eingebaute Planer nicht alle Sprachfeatures von PDDL unterstützt. Eine Fehlerausgabe kann also sowohl darauf hindeuten, dass die Problembeschreibung fehlerhaft ist, als auch darauf, dass nicht unterstützte Sprachfeatures verwendet wurden. Beachten Sie ferner, dass die Eingabe für einen Planer immer aus zwei PDDL-Dateien besteht. Die Domänendatei enthält alle Informationen, die über verschiedene Probleminstanzen derselben Domäne identisch sind (wie Prädikate, Aktionsschemata u. ä.), während die Problemdatei alle instanzspezifischen Informationen enthält (also Objekte, Ist-Zustand, Soll-Zustand u. ä.).

(b) Machen Sie sich mit KAMI bzw. KAMI 2 vertraut. Die Spielregeln sind äußerst einfach: eine Fläche ist mit verschiedenfarbigen Teilen gekachelt. Das Ziel ist es, die Teile in möglichst wenigen Schritten so umzufärben, dass am Ende alle Teile dieselbe Farbe haben. Eine Aktion besteht darin, genau einem Teil eine neue Farbe zuzuweisen. Dabei werden allerdings alle weiteren Teile ebenfalls entsprechend umgefärbt, die in derselben farblich und räumlich zusammenhängenden Komponente der Fläche wie das gewählte Teil liegen.

(c) Modellieren Sie KAMI in PDDL. Zunächst zur Domänendatei: Es gibt sicher eine Vielzahl von Modellierungsmöglichkeiten, und wir wollen Ihnen keine zu engen Vorgaben bei der Modellierung machen. Folgende PDDL-Sprachfeatures könnten sich jedoch als nützlich erweisen: abgeleitete Prädikate (*derived predicates*), quantifizierte Vorbedingungen und Effekte (*quantified preconditions and effects*), sowie bedingte Effekte (*conditional effects*).

[1] `http://www.fast-downward.org`
[2] `https://www.stateofplaygames.com/kami/`
[3] `https://www.stateofplaygames.com/kami2/`
[4] `http://icaps-conference.org/ipc2008/deterministic/PddlResources.html`
[5] `http://editor.planning.domains/`

Nun zur Problemdatei: Erstellen Sie eine Problemdatei, die neben dem Ziel, dass alle Teile dieselbe Farbe haben sollen, folgenden Initialzustand beschreibt:



(d) Lösen Sie den erstellten KAMI-Level mit FAST DOWNWARD:

```
./fast-downward.py domain.pddl problem.pddl --search "astar(blind())"
```

Mit diesem Aufruf löst FAST DOWNWARD das Problem mittels A*-Suche und der blinden Heuristik. Welchen Plan findet FAST DOWNWARD? Ist der Plan korrekt? Ist er optimal? Können Sie das Problem mit anderen Konfigurationen von FAST DOWNWARD schneller lösen? Können Sie durch Änderungen der Modellierung in PDDL die Planung beschleunigen?

(e) Die manuelle Erstellung der Problemdatei in Teilaufgabe (c) war vermutlich etwas umständlich und langweilig. Wir wollen sie deshalb automatisieren. Schreiben Sie ein Skript, welches eine PDDL-Problemdatei aus einer minimalen textuellen Beschreibung des Levels erstellt. Die textuelle Beschreibung sollte aus einem Zeichen pro Teil bestehen, das die Farbe des Teils kodiert (also etwa `R` für rot, `G` für grün, `B` für blau, `Y` für gelb usw.). Die Textdatei für das Problem aus Teilaufgabe (c) sieht beispielsweise wie folgt aus:

```
BBBBBB
BBBBBB
YYYYYY
YBBBBY
YBYYBY
YBYYBY
YBBBBY
YYYYYY
BBBBBB
BBBBBB
```

(f) Erstellen Sie ein Skript, das die obigen Arbeitsschritte zusammenfasst und automatisiert. Das Skript sollte eine Textdatei mit einer kompakten Kodierung der Probleminstanz (wie oben) als Eingabe nehmen, daraus eine PDDL-Datei erstellen, das Problem mit FAST DOWNWARD lösen, und schließlich den gefundenen Plan ausgeben.

Testen Sie das Skript mit einigen weiteren, evtl. schwierigeren Probleminstanzen. Diese können Sie entweder selbst erstellen oder von uns erhalten.

(g) Reflektieren Sie: Was haben Sie in dieser Aufgabe gelernt? Welche weiteren Optimierungen wären denkbar? Welche Optimierungen könnten in PDDL durchgeführt werden? Welche wären eher in einem spezialisierten KAMI-Löser umsetzbar?

Aufgabe 3.2 (Beschreibungslogiken)

Beschreibungslogiken sind Sprachen zur Darstellung von terminologischem Wissen. Neben der Darstellung ist auch das Schlussfolgern über solches Wissen möglich, also die Ableitung von implizitem Wissen aus einer Wissensbasis.
In dieser Aufgabe wollen wir uns mit Beschreibungslogiken sowie mit einem Ontologie-Editor vertraut machen, der das praktische Arbeiten mit Ontologien erheblich vereinfacht. Wir wollen unser gewonnenes Wissen einsetzen, um selbst eine kleine Ontologie zu modellieren und über sie zu schlussfolgern.

(a) Machen Sie sich mit Beschreibungslogiken vertraut. Wir empfehlen zum Beispiel *A Description Logic Primer* von Krötzsch, Simancík und Horrocks[6].

(b) Machen Sie sich mit dem Ontologie-Editor Protégé[7] vertraut. Installieren Sie dafür die Desktop-Version von Protégé und arbeiten Sie eines der im Protégé-Wiki[8] verlinkten Tutorials durch. Wir empfehlen für einen schnellen Start *Getting Started with Protege Desktop Editor* gefolgt von *Pizzas in 10 minutes: A demo of modeling shortcuts*, ggf. ergänzt durch *Ontology Development 101*.

Alternativ zu den schriftlichen Tutorials gibt es auch ein empfehlenswertes Youtube-Tutorial[9].

(c) Modellieren Sie in Protégé eine Beispielontologie zum Baden-Württembergischen Kommunalwahlrecht: Angenommen, es liegt eine Wissensdatenbank vor, in der alle in Baden-Württemberg wohnhaften Personen zusammen mit personenbezogenen Daten wie Geschlecht, Alter, Staatsbürgerschaft und Wohnort (nach Stadtteilen) gespeichert sind. Ferner ist repräsentiert, welche Stadtteile zu welchen Städten gehören. Modellieren Sie in Protégé die Klasse aller in Freiburg wahlberechtigten Personen (`DarfInFreiburgWaehlen`) als Klasse aller EU-Bürger, die in (einem Stadtteil von) Freiburg wohnen und die mindestens 16 Jahre alt sind.

Erstellen Sie eine Reihe von Personen-Instanzen, die alle Möglichkeiten von Wahlberechtigten und Nicht-Wahlberechtigten abdecken (also EU-Bürger und Nicht-EU-Bürger, Freiburger und Nicht-Freiburger, Über- und Unter-Sechzehnjährige). Lassen Sie die Individuen vom mitgelieferten Reasoner (HermiT) klassifizieren. Erhalten Sie das erwartete Ergebnis?

Lassen Sie sich von Protégé bzw. HermiT einige weitere Fragen beantworten: Ist jeder, der in Freiburg wählen darf, EU-Bürger? Deutscher? Freiburger? Wiehremer? Besteht Frauenwahlrecht? Dürfen Schweizer wählen? Erhalten Sie immer das erwartete Ergebnis? Wenn nicht, woran könnte dies liegen?

(d) *Bonus:* Angenommen, Sie modellierten die Informationen aus Aufgabe 1 (also Teile, deren Farben, Nachbarschaftsbeziehungen zwischen Teilen, Zusammenhangskomponenten, etc.) geeignet in Protégé. Könnten Sie sich automatisches Schlussfolgern in der erstellten Ontologie für das Lösen der Planungsaufgaben aus Aufgabe 1 zunutze machen?

[6] `https://arxiv.org/pdf/1201.4089.pdf`
[7] `https://protege.stanford.edu/`
[8] `https://protegewiki.stanford.edu/wiki/Main_Page`
[9] `https://www.youtube.com/playlist?list=PLD8uCWff9n-EG4KK2OAiPRSCPgNJXf49j`