Handlungsplanung

Prof. Dr. B. Nebel, Dr. M. Helmert
R. Mattmüller
Wintersemester 2006/2007

Universität Freiburg
Institut für Informatik

Projekt P2

Abgabe: Mittwoch, 31. Januar 2007

In diesem Projekt entwickeln Sie einen SAT-Planer, der auf den beiden Kodierungen aus der Vorlesung beruht.

Zur Vereinfachung können Sie PDDL-kodierte Planungsprobleme mit dem Tool `pddlcat`¹ vorverarbeiten. Der Aufruf

```
pddlcat -ins domain.pddl problem.pddl -pddl1 -no-dkel -p > outfile.pddl
```

schreibt eine instanziierte und vereinfachte Version des Problems `problem.pddl` aus der Domäne `domain.pddl` in `outfile.pddl`. Der Inhalt von `outfile.pddl` ist nach der Grammatik aus Aufgabe P2.1 aufgebaut.

Aufgabe P2.1 (PDDL-Eingabe – 1 Punkt)

Schreiben Sie einen Parser für die Ausgabe von `pddlcat`. Wählen Sie eine geeignete interne Repräsentation des Planungsproblems.

```
PDDL ::= DOMAIN PROBLEM
DOMAIN ::= (define (domain NAME)
            (:predicates ATOM+)
            ACTION*)
ATOM ::= (NAME)
ACTION ::= (:action NAME
            :precondition FORMULA
            :effect EFFECT)
FORMULA ::= ATOM
           | (not FORMULA)
           | (and FORMULA*)
           | (or FORMULA*)
EFFECT ::= ATOM
          | (not ATOM)
          | (and EFFECT*)
PROBLEM ::= (define (problem NAME)
            (:domain NAME)
            (:init ATOM*)
            (:goal FORMULA))
```

Die Domänendefinition enthält zunächst den Namen² der Domäne, eine Liste von instanziierten Prädikaten und eine Liste von ebenfalls instanziierten Aktionen. Vorbedingungen dürfen beliebige propositionale Formeln mit `and`, `or` und

¹Teil des HSP*-Pakets, <http://www.ida.liu.se/~pahas/hsp/>

²Namen in PDDL sind mit einem Buchstaben beginnende Zeichenketten aus Buchstaben, Ziffern, Bindestrichen und Unterstrichen, in denen nicht zwischen Groß- und Kleinschreibung unterschieden wird.

not sein. Effekte sind entweder atomare Add- oder Delete-Effekte (ATOM bzw. (not ATOM)) oder Listen von Effekten (and EFFECT*³).

Die auf die Domänenendefinition folgende Problemdefinition enthält die Namen des Problems und der Domäne sowie eine Liste der im Anfangszustand gültigen Propositionen (*closed world assumption*) und eine Zielformel.

Aufgabe P2.2 (Aussagenlogische Kodierung – 1,5 Punkte)

Schreiben Sie einen Übersetzer von dem PDDL-Fragment aus Aufgabe P2.1 in Aussagenlogik. Der Übersetzer soll die beiden in der Vorlesung vorgestellten Kodierungen Φ_t^{seq} und Φ_t^{par} beherrschen.

Transformieren Sie die entstandene Formel in konjunktive Normalform.

Numerieren Sie die vorkommenden Aussagenvariablen durch und verwalten Sie eine Symboltabelle für die Variablen.

Geben Sie die Formel im DIMACS-Format für KNF-Formeln⁴ in eine Datei aus. Die Datei enthält zu Beginn eine Zeile der Form

```
p cnf VARIABLES CLAUSES
```

mit Variablenanzahl VARIABLES und Klauselanzahl CLAUSES. Jede Klausel wird durch die Indizes ihrer Variablen (beginnend bei 1) repräsentiert. Indizes von positiven Literalen werden unverändert, Indizes von negierten Variablen negiert angegeben. Klauseln werden durch das Zeichen 0 beendet. Die Formel $(x_1 \vee x_3 \vee x_4) \wedge \neg x_4 \wedge (x_2 \vee \neg x_3)$ wird etwa repräsentiert als

```
p cnf 4 3
1 3 4 0
-4 0
2 -3 0
```

Aufgabe P2.3 (SAT-Solving, Ausgabe – 1 Punkt)

Erzeugen Sie systematisch die Formeln $\Phi_0^{seq}, \Phi_1^{seq}, \dots$ bzw. $\Phi_0^{par}, \Phi_1^{par}, \dots$ und lassen Sie sie von einem SAT-Solver lösen, bis die erste erfüllbare Formel gefunden ist.

Verwenden Sie den SAT-Solver „Siege“ (variant 4)⁵. Siege legt eine Datei mit dem Namen `siege.results` im aktuellen Arbeitsverzeichnis an, die für jeden Erfüllbarkeitstest eine Zeile der Form

```
(CNFFILENAME, unsatisfiable)
```

enthält, wenn die Formel in Datei CNFFILENAME unerfüllbar ist, bzw.

```
(CNFFILENAME, { VALUATION })
```

³Sie dürfen auch bedingte Effekte (`when FORMULA EFFECT`) berücksichtigen.

⁴Siehe <http://www.satlib.org/Benchmarks/SAT/satformat.ps>, Abschnitt 2.1

⁵Download unter <http://www.cs.sfu.ca/research/groups/CL/software/siege/>

mit einer erfüllenden Belegung VALUATION, wenn die Formel erfüllbar ist. Enthält etwa die Datei `formula.cnf` die Kodierung der Beispielformel $(x_1 \vee x_3 \vee x_4) \wedge \neg x_4 \wedge (x_2 \vee \neg x_3)$ aus der vorigen Aufgabe, so enthält `siege.results` nach dem Aufruf `siege_v4 formula.cnf` die Zeile (`formula.cnf`, { 1 2 3 -4 }), die der Belegung v mit $v(x_1) = v(x_2) = v(x_3) = 1$ und $v(x_4) = 0$ entspricht.

Übersetzen Sie erfüllende Belegungen mit Hilfe der in Aufgabe P2.2 angelegten Symboltabelle in Pläne und geben Sie die Pläne in einer Lösungsdatei aus, die nach Zeitpunkten geordnet für jede Aktion eine Zeile

TIMEPOINT: ACTION [DURATION]

enthält. Ein paralleler Plan für das Blocksworld-Problem mit vier auf dem Tisch liegenden Blöcken A, B, C und D , die in die Konfiguration A -auf- B und C -auf- D gebracht werden sollen, könnte dann kodiert werden als

```
0: (pick-up A) [1]
0: (pick-up C) [1]
1: (stack A B) [1]
1: (stack C D) [1]
```

Aufgabe P2.4 (SAT-Planer – 0,5 Punkte)

Integrieren Sie `pddlcat`, `siege_v4` und ihr(e) Programm(e) aus den Aufgaben P2.1 bis P2.3 zu einem Programm `satplan_p2`, das als

```
satplan_p2 encoding domain.pddl problem.pddl solution.soln
```

aufgerufen werden kann, Kodierung `encoding` $\in$ {seq,par} verwendet und einen Plan für `domain.pddl` und `problem.pddl` in Datei `solution.soln` schreibt. Optional kann die Lösungsdatei führende Kommentarzeilen mit Metainformationen über benötigte Zeit, Anzahl der Aktionen, Anzahl der Planschritte oder verwendete Kodierung enthalten (vgl. in Projekt 1 erzeugte Lösungsdateien).

Das Projekt darf und sollte in Gruppen von zwei Studenten bearbeitet werden. Geben Sie in Ihrer Lösung beide Namen an.

Geben Sie eine komprimierte Archiv-Datei ab, die alle von Ihnen geschriebenen Dateien (nur Quellen, keine Binaries, ggf. zusammen mit einem Makefile o.ä.) enthält. Wenn Sie den PDDL-Parser mit Hilfe eines Parser-Generators erzeugen, geben Sie bitte sowohl die Eingabe des Parser-Generators als auch die von ihm erzeugten Quell-Dateien ab.