

Grundlagen der Künstlichen Intelligenz

31. Handlungsplanung: Planungsformalismen

Malte Helmert

Universität Basel

5. Mai 2014

Grundlagen der Künstlichen Intelligenz

5. Mai 2014 — 31. Handlungsplanung: Planungsformalismen

31.1 Vier Formalismen

31.2 STRIPS

31.3 ADL, SAS⁺ und PDDL

31.4 Zusammenfassung

Handlungsplanung: Überblick

Kapitelüberblick:

- ▶ 30. Einführung
- ▶ 31. Planungsformalismen
- ▶ 32.-37. Planungsheuristiken

31.1 Vier Formalismen

Vier Planungsformalismen

- ▶ Eine Beschreibungssprache für Zustandsräume (**Planungsaufgaben**) nennt man **Planungsformalismus**.
- ▶ Wir stellen vier Planungsformalismen vor:
 - ① STRIPS (von **Stanford Research Institute Problem Solver**)
 - ② ADL (**Action Description Language**)
 - ③ SAS⁺ (**Simplified Action Structures**)
 - ④ PDDL (**Planning Domain Definition Language**)
- ▶ STRIPS und SAS⁺ sind die einfachsten Formalismen; in den Folgekapiteln werden wir uns auf sie beschränken.

31.2 STRIPS

Vier Planungsformalismen: STRIPS

Wir stellen vier Planungsformalismen vor:

- ① **STRIPS**
- ② ADL
- ③ SAS⁺
- ④ PDDL

STRIPS: Grundkonzepte

Grundkonzepte von STRIPS:

- ▶ STRIPS ist der **einfachste** übliche Planungsformalismus
- ▶ Zustandsvariablen sind **binär** (wahr oder falsch)
- ▶ **Zustände** s (gegeben Zustandsvariablen V) können auf zwei äquivalente Arten beschrieben werden:
 - ▶ als **Belegungen** $s : V \rightarrow \{\mathbf{F}, \mathbf{T}\}$
 - ▶ als **Mengen** $s \subseteq V$, wobei die Menge der in s **wahren** Zustandsvariablen kodiert wird

Wir verwenden die Mengenschreibweise, da sie praktischer ist.
- ▶ **Zielzustände** und **Vorbedingungen von Aktionen** gegeben durch Menge von Variablen, die **wahr** sein müssen (Werte der anderen Variablen im Ziel sind egal)
- ▶ **Effekte von Aktionen** gegeben durch Angabe von Variablen, die **wahr werden** bzw. **falsch werden**

STRIPS-Planungsaufgabe

Definition (STRIPS-Planungsaufgabe)

Eine **STRIPS**-Planungsaufgabe ist ein 4-Tupel $\Pi = \langle V, I, G, A \rangle$ mit folgenden Komponenten:

- ▶ V : endliche Menge von **Zustandsvariablen**
- ▶ $I \subseteq V$: der **Anfangszustand**
- ▶ $G \subseteq V$: die Menge der **Ziele**
- ▶ A : endliche Menge von **Aktionen**, wobei für jede Aktion $a \in A$ definiert sind:
 - ▶ $pre(a) \subseteq V$: ihre **Vorbedingungen**
 - ▶ $add(a) \subseteq V$: ihre **Add-Effekte**
 - ▶ $del(a) \subseteq V$: ihre **Delete-Effekte**
 - ▶ $cost(a) \in \mathbb{N}_0$: ihre **Kosten**

Anmerkung: Aktionskosten sind eine Erweiterung gegenüber „traditionellem“ STRIPS

Zustandsraum zu einer STRIPS-Planungsaufgabe

Definition (von STRIPS-Planungsaufgabe induz. Zustandsraum)

Sei $\Pi = \langle V, I, G, A \rangle$ eine STRIPS-Planungsaufgabe.

Dann **induziert** Π den **Zustandsraum** $\mathcal{S}(\Pi) = \langle S, A, cost, T, s_0, S_\star \rangle$:

- ▶ **Zustandsmenge:** $S = 2^V$ (= Potenzmenge von V)
- ▶ **Aktionen:** die Aktionen A von Π
- ▶ **Aktionskosten:** $cost$ ist wie in Π definiert
- ▶ **Transitionen:** $s \xrightarrow{a} s'$ für Zustände s, s' und Aktion a gdw.:
 - ▶ $pre(a) \subseteq s$ (Vorbedingungen erfüllt)
 - ▶ $s' = (s \setminus del(a)) \cup add(a)$ (Effekte werden angewandt)
- ▶ **Anfangszustand:** $s_0 = I$
- ▶ **Zielzustände:** $s \in S_\star$ für Zustand s gdw. $G \subseteq s$ (Ziele erreicht)

Beispiel: Blocks world in STRIPS

Beispiel (eine Blocks-world-Planungsaufgabe in STRIPS)

$\Pi = \langle V, I, G, A \rangle$ mit:

- ▶ $V = \{on_{A,B}, on_{A,C}, on_{B,A}, on_{B,C}, on_{C,A}, on_{C,B}, on_table_A, on_table_B, on_table_C, clear_A, clear_B, clear_C\}$
- ▶ $I = \{on_{C,A}, on_table_A, on_table_B, clear_C, clear_B\}$
- ▶ $G = \{on_{A,B}, on_{B,C}\}$
- ▶ $A = \{move_{A,B,C}, move_{A,C,B}, move_{B,A,C}, move_{B,C,A}, move_{C,A,B}, move_{C,B,A}, to_table_{A,B}, to_table_{A,C}, to_table_{B,A}, to_table_{B,C}, to_table_{C,A}, to_table_{C,B}, from_table_{A,B}, from_table_{A,C}, from_table_{B,A}, from_table_{B,C}, from_table_{C,A}, from_table_{C,B}\}$

...

Beispiel: Blocks world in STRIPS

Beispiel (eine Blocks-world-Planungsaufgabe in STRIPS)

move-Aktionen kodieren Bewegung eines Blocks von einem Block auf einen anderen

Beispielhaft:

- ▶ $pre(move_{A,B,C}) = \{on_{A,B}, clear_A, clear_C\}$
- ▶ $add(move_{A,B,C}) = \{on_{A,C}, clear_B\}$
- ▶ $del(move_{A,B,C}) = \{on_{A,B}, clear_C\}$
- ▶ $cost(move_{A,B,C}) = 1$

Beispiel: Blocks world in STRIPS

Beispiel (eine Blocks-world-Planungsaufgabe in STRIPS)

to-table-Aktionen kodieren Bewegung eines Blocks von einem Block auf den Tisch

Beispielhaft:

- ▶ $pre(to-table_{A,B}) = \{on_{A,B}, clear_A\}$
- ▶ $add(to-table_{A,B}) = \{on-table_A, clear_B\}$
- ▶ $del(to-table_{A,B}) = \{on_{A,B}\}$
- ▶ $cost(to-table_{A,B}) = 1$

Beispiel: Blocks world in STRIPS

Beispiel (eine Blocks-world-Planungsaufgabe in STRIPS)

from-table-Aktionen kodieren Bewegung eines Blocks vom Tisch auf einen Block

Beispielhaft:

- ▶ $pre(from-table_{A,B}) = \{on-table_A, clear_A, clear_B\}$
- ▶ $add(from-table_{A,B}) = \{on_{A,B}\}$
- ▶ $del(from-table_{A,B}) = \{on-table_A, clear_B\}$
- ▶ $cost(from-table_{A,B}) = 1$

Warum STRIPS?

- ▶ STRIPS ist **besonders einfach**
- ↪ erleichtert Entwurf und Implementierung von Planungsalgorithmen
- ▶ oft umständlich für den „Anwender“, Probleme direkt in STRIPS zu formulieren
- ▶ **aber:** STRIPS ist genauso „mächtig“ wie sehr viel komplexere Planungssprachen
- ↪ automatische „Compiler“ von komplexeren Sprachen (wie ADL und SAS⁺) nach STRIPS existieren

31.3 ADL, SAS⁺ und PDDL

Vier Planungsformalismen: ADL, SAS⁺ und PDDL

Wir stellen vier Planungsformalismen vor:

- 1 STRIPS
- 2 ADL
- 3 SAS⁺
- 4 PDDL

ADL

Grundkonzepte von ADL:

- ▶ ADL verwendet wie STRIPS Aussagevariablen (wahr/falsch) als Zustandsvariablen
- ▶ Vorbedingungen von Aktion und Ziel können **beliebige logische Formeln** sein (Aktion anwendbar/Ziel erreicht in Zuständen, die die Formel erfüllen)
- ▶ neben STRIPS-Effekten gibt es **bedingte Effekte**: Variable v wird nur auf falsch/wahr gesetzt, wenn gegebene logische Formel im aktuellen Zustand erfüllt ist

Grundkonzepte von SAS⁺

Grundkonzepte von SAS⁺:

- ▶ sehr ähnlich zu STRIPS, aber Zustandsvariablen nicht binär, sondern mit gegebenen **endlichen Wertebereichen** (vgl. CSPs)
- ▶ Zustände sind **Belegungen** dieser Variablen (wie bei CSPs)
- ▶ Vorbedingungen und Ziele über **partielle Belegungen** gegeben

Beispiel: $\{v_1 \mapsto a, v_3 \mapsto b\}$ als Vorbedingung bzw. Ziel

- ▶ Wenn im Zustand s gilt: $s(v_1) = a$ und $s(v_3) = b$, dann ist Aktion anwendbar bzw. Ziel erreicht.
- ▶ Werte anderer Variablen sind egal.

- ▶ Effekte sind **Zuweisungen an Teilmenge** der Variablen

Beispiel: Effekt $\{v_1 \mapsto b, v_2 \mapsto c\}$ bedeutet

- ▶ Im Nachfolgezustand s' gilt $s'(v_1) = b$ und $s'(v_2) = c$.
- ▶ Alle anderen Variablen behalten ihren Wert bei.

Grundkonzepte von PDDL

- ▶ PDDL ist die in der Praxis verwendete Standardsprache für Planungsaufgaben.
- ▶ Beschreibungen in (eingeschränkter) Prädikatenlogik statt Aussagenlogik (eine weitere Stufe kompakter)
- ▶ weitere Features, z.B. **numerische Variablen** und **abgeleitete Variablen (Axiome)** zur Definition von „Makros“ (Formeln, die in jedem Zustand automatisch neu ausgewertet werden und z. B. in Vorbedingungen verwendet werden können)
- ▶ Es gibt definierte STRIPS- und ADL-Fragmente; sehr viele Planer unterstützen nur das STRIPS-Fragment

Beispiel: Blocks world in PDDL

31.4 Zusammenfassung

Zusammenfassung

Planungsformalismen:

- ▶ **STRIPS**: besonders einfach, daher für Algorithmen angenehm
 - ▶ binäre Zustandsvariablen
 - ▶ Vorbedingungen, Add- und Delete-Effekte, Ziele: Mengen von Variablen
- ▶ **ADL**: Erweiterung von STRIPS
 - ▶ **Logikformeln** für komplexe Vorbedingungen und Ziele
 - ▶ **bedingte Effekte**
- ▶ **SAS⁺**: Erweiterung von STRIPS
 - ▶ Zustandsvariablen mit **beliebigen endlichen Wertebereichen**
- ▶ **PDDL**: praktisch verwendete Eingabesprache
 - ▶ basiert auf Prädikatenlogik (kompakter als Aussagenlogik)
 - ▶ von den meisten Algorithmen nur teilweise unterstützt (z.B. STRIPS- oder ADL-Fragment)