Grundlagen der Künstlichen Intelligenz

14. Handlungsplanung: Einführung

Malte Helmert

Universität Basel

3. Mai 2013

Einordnung

Einordnung:

Handlungsplanung

Umgebung:

- statisch vs. dynamisch
- deterministisch vs. nicht-deterministisch vs. stochastisch
- vollständig vs. partiell vs. nicht beobachtbar
- diskret vs. stetig
- ein Agent vs. mehrere Agenten

Lösungsansatz:

- problemspezifisch vs. allgemein vs. lernend

Einführung

Handlungsplanung

Was ist Handlungsplanung?

„Planning is the art and practice of thinking before acting.“
— P. Haslum

$\rightsquigarrow$ Finden von Plänen (Aktionsfolgen), die von einem Anfangszustand aus einen Zielzustand erreichen

Unser Thema ist klassische Handlungsplanung:

- allgemeiner Ansatz für Lösung von „Suchproblemen“ (aus Kapiteln 3–5)
- klassisch = statisch, deterministisch, vollständig beobachtbar
- Varianten: probabilistisches Planen, Planen mit partieller Beobachtbarkeit, Online-Planen, ...

Handlungsplanung informell

Gegeben:

- Beschreibung eines Zustandsraums in einer geeigneten Beschreibungssprache (Planungsformalismus)

Gesucht:

- ein Plan, d. h. Lösung für den beschriebenen Zustandsraum (Folge von Aktionen vom Anfangszustand zum Ziel)
- oder ein Nachweis, dass kein Plan existiert

Unterscheidung zwischen

- optimalen Planern: garantieren, dass gefundene Pläne optimal sind, d. h. minimale Gesamtkosten haben
- suboptimalen Planern (satisficing): dürfen suboptimale Pläne liefern

Was ist neu?

Viele Probleme, die wir schon kennen, sind im Prinzip Planungsprobleme:

- Blocks world
- Missionare und Kannibalen
- 15-Puzzle

Neu ist, dass wir uns nun für allgemeine Algorithmen interessieren, d. h. der Entwickler des Suchalgorithmus kennt nicht die zu lösenden Probleme.

- $\rightsquigarrow$ keine problemspezifischen Heuristiken!
- $\rightsquigarrow$ Eingabesprache, die zu lösende Probleme modelliert

Handlungsplanung: Überblick

Kapitelüberblick:

- Einführung und Formalisierung $\rightsquigarrow$ dieses Kapitel
- Delete-Relaxierung $\rightsquigarrow$ Kapitel 15
- Abstraktion $\rightsquigarrow$ Kapitel 16
- Landmarken $\rightsquigarrow$ Kapitel 17

Wiederholung: Zustandsräume

Über diesen Abschnitt

Nichts neues hier!

Dieser Abschnitt ist eine Wiederholung
von Abschnitt 3.2 und Teilen von Abschnitt 3.3
aus dem Kapitel „Klassische Suchprobleme“.

Formalisierung von Zustandsräumen

Vorbemerkungen:

- Um Suchprobleme sauber algorithmisch fassen zu können, benötigen wir eine formale Definition.
- grundlegendes semantisches Konzept: Zustandsräume
- Zustandsräume sind (annotierte) Graphen
- Pfade zu Zielzuständen entsprechen Lösungen
- kürzeste Pfade entsprechen optimalen Lösungen

Zustandsräume

Definition (Zustandsraum)

Ein Zustandsraum ist ein 6-Tupel $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ mit

- S endliche Menge von Zuständen
- A endliche Menge von Aktionen
- $cost : A \to \mathbb{R}_0^+$ Aktionskosten
- $T \subseteq S \times A \times S$ Transitionsrelation oder Übergangsrelation; deterministisch in $\langle s, a \rangle$ (siehe nächste Folie)
- $s_0 \in S$ Anfangszustand
- $S_\star \subseteq S$ Menge der Zielzustände

- auch: Transitionssystem (transition system)
- englisch: state space, state, action, action costs, transition relation, initial state, goal states

Zustandsräume: Transitionen, Determinismus

Definition (Transition, deterministisch)

Sei $\mathcal{S} = \langle S, A, \mathit{cost}, T, s_0, S_\star \rangle$ ein Zustandsraum.

Die Tripel $\langle s, a, s' \rangle \in T$ heissen Transitionen/Zustandsübergänge.

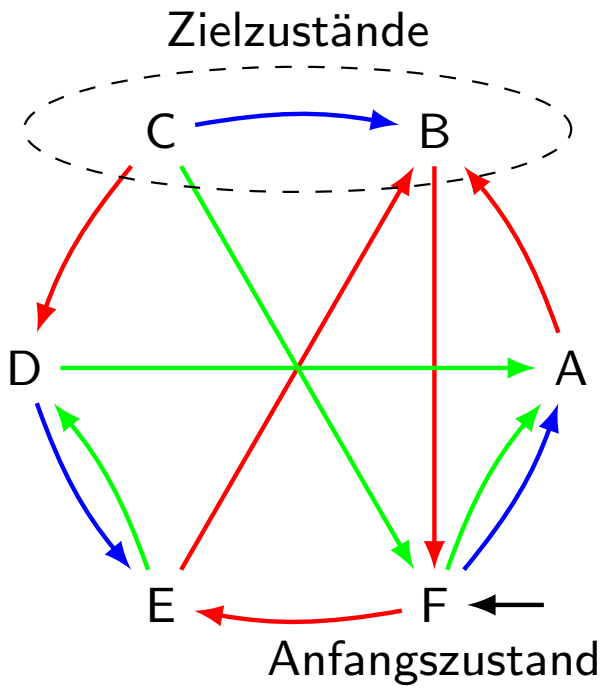
Wir sagen $\mathcal{S}$ hat die Transition $\langle s, a, s' \rangle$, falls $\langle s, a, s' \rangle \in T$ und schreiben dafür $s \xrightarrow{a} s'$ sowie $s \to s'$, wenn a nicht interessiert.

Transitionen sind deterministisch in $\langle s, a \rangle$: $s \xrightarrow{a} s_1$ und $s \xrightarrow{a} s_2$ mit $s_1 \neq s_2$ ist nicht erlaubt.

Zustandsraum: Beispiel

Zustandsräume werden oft als gerichtete Graphen dargestellt.

- Zustände: Graphknoten
- Transitionen: beschriftete Kanten
  (hier: Färbung statt Beschriftung)
- Anfangszustand: eingehender Pfeil
- Zielzustände: markiert
  (hier: durch Ellipse)
- Aktionen: die Kantenbeschriftungen
- Aktionskosten: separat anzugeben
  (oder implizit $= 1$)



Zustandsräume: Begriffe

Wir verwenden übliche Begriffe aus der Graphentheorie.

Definition (Vorgänger, Nachfolger, anwendbare Aktionen)

Sei $\mathcal{S} = \langle S, A, \mathit{cost}, T, s_0, S_\star \rangle$ ein Zustandsraum.

Seien $s, s' \in S$ Zustände mit $s \to s'$.

- s ist Vorgänger von s'
- s' ist Nachfolger von s

Wenn $s \xrightarrow{a} s'$ gilt, ist die Aktion a anwendbar in s.

Zustandsräume: Begriffe

Wir verwenden übliche Begriffe aus der Graphentheorie.

Definition (Pfad)

Sei $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ ein Zustandsraum.

Seien $s^{(0)}, \ldots, s^{(n)} \in S$ Zustände und $\pi_1, \ldots, \pi_n \in A$ Aktionen, so dass $s^{(0)} \xrightarrow{\pi_1} s^{(1)}, \ldots, s^{(n-1)} \xrightarrow{\pi_n} s^{(n)}$.

- $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ ist Pfad von $s^{(0)}$ nach $s^{(n)}$
- Länge des Pfads: $|\pi| = n$
- Kosten des Pfads: $cost(\pi) = \sum_{i=1}^{n} cost(\pi_i)$

Anmerkungen:

- Pfade der Länge 0 sind erlaubt
- manchmal nennt man auch die Zustandsfolge $\langle s^{(0)}, \ldots, s^{(n)} \rangle$ oder die Folge $\langle s^{(0)}, \pi_1, s^{(1)}, \ldots, s^{(n-1)}, \pi_n, s^{(n)} \rangle$ Pfad

Zustandsräume: Begriffe

Weitere Begriffe:

Definition (Lösung, optimal, lösbar, erreichbar, Sackgasse)

Sei $\mathcal{S} = \langle S, A, cost, T, s_0, S_\star \rangle$ ein Zustandsraum.

- Ein Pfad von einem Zustand $s \in S$ zu einem Zustand $s_\star \in S_\star$ ist Lösung für/von s.
- Eine Lösung für s_0 ist Lösung für/von $\mathcal{S}$.
- Optimale Lösungen (für s) haben minimale Kosten unter allen Lösungen (für s).
- Zustandsraum $\mathcal{S}$ ist lösbar, wenn eine Lösung für $\mathcal{S}$ existiert.
- Zustand s ist erreichbar, wenn Pfad von s_0 zu s existiert.
- Zustand s ist Sackgasse, wenn keine Lösung für s existiert.

Repräsentation von Zustandsräumen

Wie kommt der Zustandsraum in den Computer?

vorher: als Black Box

jetzt: als deklarative Beschreibung

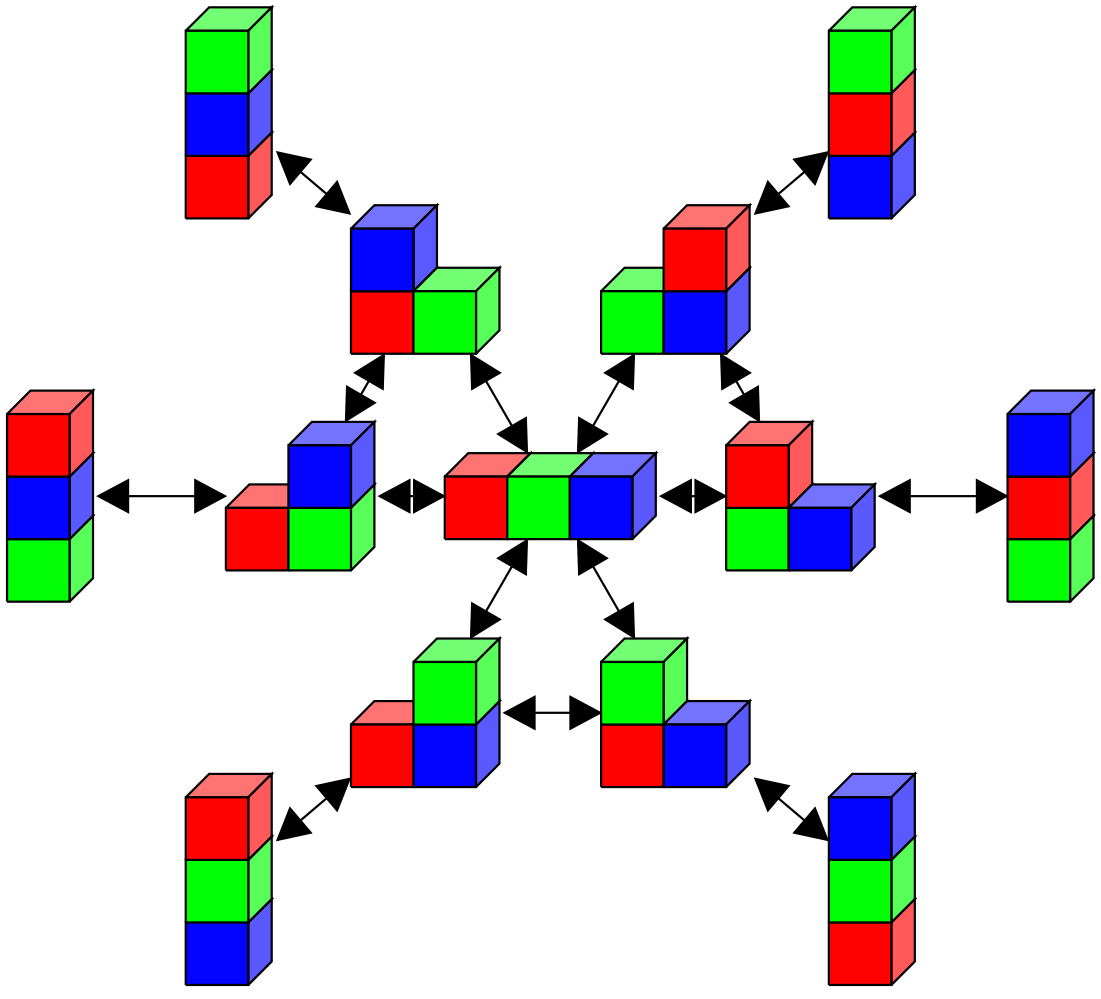
- kompakte Beschreibung des Zustandsraums als Input
  $\rightsquigarrow$ Zustandsraum exponentiell grösser als Input in Bytes
- Lösungsverfahren arbeiten direkt mit kompakter Beschreibung
  (können z. B. Problem automatisch umformulieren/vereinfachen)

Kompakte Beschreibungen

Kompakte Beschreibungen

- In diesem Abschnitt beschreiben wir beispielhaft, wie eine kompakte deklarative Beschreibung eines Zustandsraums funktionieren kann.
- Im folgenden Abschnitt geben wir dann konkrete Details für praktisch verwendete Beschreibungssprachen.

Erinnerung: Blocks world



Kompakte Beschreibung von Zustandsräumen

Wie können wir Zustandsräume kompakt beschreiben?

Kompakte Beschreibung von vielen Zuständen

- Führe Zustandsvariablen ein.
- Zustände sind Zuweisungen an die Zustandsvariablen.

$\rightsquigarrow$ z. B. n binäre Zustandsvariablen können 2^n Zustände beschreiben

Kompakte Beschreibung von Zustandsräumen

Wie können wir Zustandsräume kompakt beschreiben?

Kompakte Beschreibung von vielen Zuständen: Blocks world

- für je zwei Blöcke b, b' eine Variable $on_{b,b'}$
- für jeden Block b eine Variable $on\text{-}table_b$
- für jeden Block b eine Variable $clear_b$

Jede Variable kann „wahr“ oder „falsch“ sein.

bei n Blöcken: $n^2 + 2n$ Variablen
beschreiben mehr als $n!$ erreichbare Zustände

Kompakte Beschreibung von Zustandsräumen

Wie können wir Zustandsräume kompakt beschreiben?

Kompakte Beschreibung von vielen Transitionen

Beschreibe für jede Aktion anhand der Zustandsvariablen,

- in welchen Zuständen sie anwendbar ist (Vorbedingung)
- und wie sie welche Zustandsvariablen ändert (Effekt)

$\rightsquigarrow$ Details unterscheiden sich je nach Planungsformalismus

Kompakte Beschreibung von Zustandsräumen

Wie können wir Zustandsräume kompakt beschreiben?

Kompakte Beschreibung von vielen Transitionen: Blocks world

Beispiel für Aktion move A from B to C:

- Vorbedingung: im aktuellen Zustand müssen $on_{A,B}$, $clear_A$ und $clear_C$ den Wert „wahr“ haben
- Effekt:
  - $on_{A,C}$ wird wahr
  - $on_{A,B}$ wird falsch
  - $clear_B$ wird wahr
  - $clear_C$ wird falsch

bei n Blöcken: $O(n^3)$ Aktionen

beschreiben mehr als $n!$ viele Transitionen

Kompakte Beschreibung von Zustandsräumen

Wie können wir Zustandsräume kompakt beschreiben?

Kompakte Beschreibung von vielen Zielzuständen

Beschreibe Menge der Zielzustände zum Beispiel

- durch separate Angabe eines Zielwerts für jede Zustandsvariable, deren Wert im Ziel wichtig ist, oder
- durch eine logische Formel über den Zustandsvariablen, die im Ziel erfüllt sein soll

$\rightsquigarrow$ Details unterscheiden sich je nach Planungsformalismus

Kompakte Beschreibung von Zustandsräumen

Wie können wir Zustandsräume kompakt beschreiben?

Kompakte Beschreibung von vielen Zielzuständen: Blocks world

Beispiel für Blocks world mit 26 Blöcken A, B, ..., Z:

- $clear_B$ soll wahr sein
- $on_{B,A}$ soll wahr sein
- $on_{A,S}$ soll wahr sein
- $on_{S,E}$ soll wahr sein
- $on_{E,L}$ soll wahr sein
- $on\text{-}table_L$ soll wahr sein
- alles andere ist egal

bei n Blöcken: mehr als 21! Zielzustände sehr kompakt beschrieben

Planungsformalismen

Vier Planungsformalismen

- Eine Beschreibungssprache für Zustandsräume (Planungsaufgaben) nennt man Planungsformalismus.
- Wir stellen im Folgenden 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.

Vier Planungsformalismen: STRIPS

Wir stellen im Folgenden 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)

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 $s : V \to \{\mathbf{F}, \mathbf{T}\}$
  - als Mengen $s \subseteq V$, wobei die Menge der in s wahren Zustandsvariablen kodiert wird

  Wir verwenden die Mengenschreibweise, da sie praktischer ist.

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 $s : V \to \{\mathbf{F}, \mathbf{T}\}$
  - 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 $\mathcal{S}(\Pi) = \langle S, A, \mathit{cost}, T, s_0, S_\star \rangle$:

- 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.:
  - $\mathit{pre}(a) \subseteq s$ (Vorbedingungen erfüllt)
  - $s' = (s \setminus \mathit{del}(a)) \cup \mathit{add}(a)$ (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:

- $V = \{on_{A,B}, on_{A,C}, on_{B,A}, on_{B,C}, on_{C,A}, on_{C,A},$
  $on\text{-}table_A, on\text{-}table_B, on\text{-}table_C,$
  $clear_A, clear_B, clear_C\}$
- $I = \{on_{C,A}, on\text{-}table_A, on\text{-}table_B, clear_C, clear_B\}$
- $G = \{on_{A,B}, on_{B,C}\}$
- $A = \{move_{A,B,C}, move_{A,C,B}, move_{B,A,C},$
  $move_{B,C,A}, move_{C,A,B}, move_{C,B,A},$
  $to\text{-}table_{A,B}, to\text{-}table_{A,C}, to\text{-}table_{B,A},$
  $to\text{-}table_{B,C}, to\text{-}table_{C,A}, to\text{-}table_{C,B},$
  $from\text{-}table_{A,B}, from\text{-}table_{A,C}, from\text{-}table_{B,A},$
  $from\text{-}table_{B,C}, from\text{-}table_{C,A}, from\text{-}table_{C,B}\}$

...

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:

- $pre(\textit{to-table}_{A,B}) = \{on_{A,B}, clear_A\}$
- $add(\textit{to-table}_{A,B}) = \{\textit{on-table}_A, clear_B\}$
- $del(\textit{to-table}_{A,B}) = \{on_{A,B}\}$
- $cost(\textit{to-table}_{A,B}) = 1$

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:

- $pre(\textit{from-table}_{A,B}) = \{\textit{on-table}_A, \textit{clear}_A, \textit{clear}_B\}$
- $add(\textit{from-table}_{A,B}) = \{on_{A,B}\}$
- $del(\textit{from-table}_{A,B}) = \{\textit{on-table}_A, \textit{clear}_B\}$
- $cost(\textit{from-table}_{A,B}) = 1$

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

Vier Planungsformalismen: ADL

Wir stellen im Folgenden 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

Vier Planungsformalismen: SAS$^+$

Wir stellen im Folgenden vier Planungsformalismen vor:

1. STRIPS
2. ADL
3. SAS$^+$
4. PDDL

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)

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.

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.

Vier Planungsformalismen: PDDL

Wir stellen im Folgenden vier Planungsformalismen vor:

1. STRIPS
2. ADL
3. SAS$^+$
4. PDDL

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-Teilfragmente; sehr viele Planer unterstützen nur das STRIPS-Fragment

Beispiel: Blocks world in PDDL