DAI-Labor
TU Berlin

Grundlagen der Künstlichen Intelligenz

Planen

10.11.2005

Brijnesh J Jain, Stefan Fricke
bjj@dai-labor.de stefan.fricke@dai-labor.de

A|O|T
Agententechnologien in betrieblichen Anwendungen und der Telekommunikation

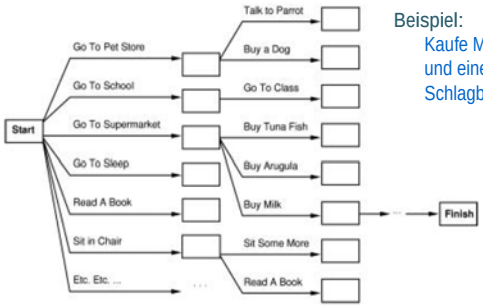
Gliederung

- Einführung
- Klassische Planungssprache – STRIPS
- Beispiele – STRIPS
- Planungsalgorithmen
- Menschliches Problemlösen (nach VL von Dr. Peter Geibel)
- Zusammenfassung

Einführung — Was ist Planen?

- Planung beschäftigt sich mit der Formalisierung, Implementierung und Evaluierung von Algorithmen für die Konstruktion von Handlungsplänen.
- Plan:
  Folge von Aktionen, deren Ausführung einen Startzustand in einen vordefinierten Zielzustand überführt.
- Das sieht nach Problemlösen aus. Was ist der Unterschied?

Einführung — Von Problemlösen zum Planen



Beispiel:
Kaufe Milch, Bananen und einen Schlagbohrer

Einführung — Von Problemlösen zum Planen

- Problemraum: vollständige Zustandsbeschreibung.
- Aktionen generieren vollständige Zustandsbeschreibung.
- Probleme:
  - Zu viele Zustände sind zu betrachten.
  - Zu viele Aktionen sind zu betrachten, hoher Verzweigungsgrad führt zu schlechter Skalierung.
  - Irrelevante Aktionen können nicht eliminiert werden.

Einführung — Von Problemlösen zum Planen

- Frage: Wie löst man hoch skalierte Probleme?
- Idee: (Kernidee des Planens):
  „*Öffne*“ Repräsentationen von Zielen, Zuständen und Aktionen
  - Ziele und Zustände als Aussagen
  - Aktionen als logische Beschreibungen mit Vorbedingungen und Effekten
  - Direkte Verbindung zwischen Aktion und Zustand

Einführung — Komponenten eines Planungssystems

- Sprache: Repräsentiert Zustände, Ziele und Aktionen
- Algorithmus: Erstellt einen Plan
- Anforderungen an Planungssysteme:
  - Genügend ausdruckstark um möglichst viele Probleme zu formalisieren
  - Genügend restriktiv, um möglichst effiziente Algorithmen anzuwenden
  - Planungsalgorithmus sollte logische Struktur des Problems ausnutzen

Gliederung

Klassische Planungssprache – STRIPS

- STRIPS = Stanford Research Institute Problem Solver (Fikes & Nilsson, 1971)
  - Mengenbasiertes Planen (auf Mengen von Prädikaten)
  - Klassischer Ansatz für Planungssysteme
  - Grundprinzipien noch heute aktuell
  - Restriktiver Formalismus; basiert auf Closed-World Assumption

STRIPS: Blockswelt — Grundlagen

- Klassische Planungsdomäne
- Hier: Zur Illustration elementarer Definitionen
- Eigenschaften der Blockswelt
  - Menge von Blöcken, die auf einem Tisch liegen
  - Blöcke können gestapelt werden
  - Es liegt maximal ein Block *direkt* auf einem anderen Block

Klassische Planungssprache – STRIPS — Grundlagen

Logische Sprache ist Tripel $L = (P, C, V)$ bestehend aus

- P : Menge von Prädikatensymbolen
  - Jedes Prädikatensymbol besitzt eine Arität (Stelligkeit)
  - Beispiel: $\{ \text{ontable}^1, \text{clear}^1, \text{on}^2 \}$
- C : Menge von Konstantensymbolen
  - Beispiel: $\{A, B, C, D\}$
- V : Menge von Variablensymbolen
  - Beispiel: $\{x, y, z, \ldots\}$
- Logische Konnektoren: $\neg$ und $\wedge$

Klassische Planungssprache – STRIPS — Grundlagen

- Terme: Konstanten und Variablen
- Atome: Ausdrücke der Form $p(t_1, \ldots, t_n)$, wobei
  - p = n-stelliges Prädikatensymbol
  - t_i = Term für alle $1 \leq i \leq n$
  - Beispiel: $\text{ontable}(A)$, $\text{clear}(x)$, $\text{on}(B,C)$
- Grundatome: Atome, die keine Variablen enthalten
  - Beispiel: $\text{ontable}(A)$, $\text{on}(B,C)$
- Literale: Atome oder negierte Atome

STRIPS: Zustände

Repräsentation

- Konjunktion von positiven Grundatomen
- Schreibweise: $p_1 \wedge p_2 \wedge \ldots \wedge p_n = \{ p_1, p_2, \ldots, p_n \}$
- Aufgrund der Closed World Assumption werden nicht aufgeführte Literale als **false** aufgefasst
- Beispiel:
  - S = { ontable(A), ontable(C), ontable(D), clear(A), clear(B), clear(D), on(B,C) }

STRIPS: Ziele

Repräsentation

- Konjunktion von positiven Grundatomen
- Typischerweise als partielle Zustandsbeschreibung
- Ein Zustand S ist ein Zielzustand Z genau dann, wenn $Z \subseteq S$
- Beispiel:
  - Z = { ontable(C), on(B,C) }

STRIPS: Aktionen

Repräsentation

- Ein Aktionsschema wird spezifiziert durch Vorbedingungen (PRE) und Effekte (EFF)
- EFFekte werden repräsentiert durch ADD und DEL Listen
- PRE, ADD sind Konjunktionen von *positiven Literalen*
- DEL ist Konjunktion von *negativen Literalen*
- Variablen werden als ∃-quantifiziert angenommen
- Eine Aktion ist ein instantiiertes Aktionsschema

Klassische Planungssprache – STRIPS

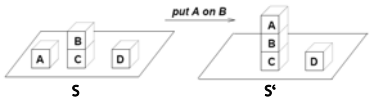
Repräsentation

Beispiel:

- ACT: put(x,y)
  - PRE: ontable(x), clear(x), clear(y)
  - ADD: on(x,y)
  - DEL: ontable(x), clear(y)

Fragen:

- Ist put(x,y) auf S anwendbar?
- Wie wird put(x,y) auf S angewendet?



STRIPS: Anwendbarkeit von Aktionen

Semantik

- Ein Aktionsschema **A** ist anwendbar für einen Zustand **S**, wenn **S** die Vorbedingung PRE von **A** erfüllt.
- Die Anwendbarkeit von **A** für Zustand **S** wird durch Substitution **θ** der Variablen in PRE überprüft:

  Substituiere alle Variablen von **A** durch Konstantensymbole

  S erfüllt PRE von **A**, wenn $\theta PRE \subseteq S$

STRIPS: Substitution

Semantik

- Substituiere Variablen mit unterschiedlichen Namen mit unterschiedlichen Konstantensymbolen.
- Substituiere alle Variablen des Aktionsschemas,
  - d.h. auch Variablen in ADD und DEL.
- Ein vollständig instantiiertes Aktionsschema ist eine Aktion **a**
  - PRE(a), ADD(a), DEL(a) bezeichnen die jeweiligen Teilmengen von a

STRIPS: Anwendbarkeit von Aktionen

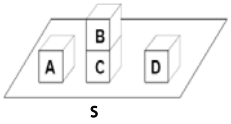
- Beispiel: Aktion put(x,y) ist anwendbar auf Zustand S

  S = { ontable(A), ontable(C), ontable(D), clear(A), clear(B), clear(D), on(B,C) } erfüllt die Vorbedingung

  PRE: ontable(x), clear(x), clear(y)

  von Aktion put(x,y) mit Substitution

  θ = {x ← A, y ← B}



S

ACT: put(x,y)
PRE: ontable(x), clear(x), clear(y)
ADD: on(x,y)
DEL: ontable(x), clear(y)

STRIPS: Anwendung von Aktionen

- Sei **S** ein Zustand und **a** ein vollständig instantiiertes Aktionsschema mit PRE(a) ⊆ S
- Das Resultat einer Aktion **a,** ausgeführt auf einen Zustand **S,** ist ein Zustand **S'** mit

$$S' = S \cup ADD(a) - DEL(a)$$

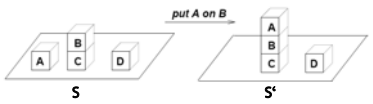
- STRIPS Annahme:
  - Jedes nicht in ADD/DEL enthaltene Literal bleibt unverändert

STRIPS: Aktionsbeispiel

Aktion put(A,B) angewendet auf Zustand

S = { ontable(A), ontable(C), ontable(D), clear(A), clear(B), clear(D), on(B,C) } liefert Zustand

S' = {ontable(C), ontable(D), clear(A), clear(D), on(B,C); on(A,B) }



S S'

ACT: put(A,B)
PRE: ontable(A), clear(A), clear(B)
ADD: on(A,B)
DEL: ontable(A), clear(B)

STRIPS: Formalisierung des Planungsproblems

- STRIPS Planungsproblem: Tripel $(A, Z, \mathcal{A})$ mit
  - A = Anfangszustand (Konjunktion von Literalen)
  - Z = Zielzustand (Konjunktion von Literalen)
  - $\mathcal{A}$ = Menge von Aktionsschemata
- Plan:
  - Eine Folge von Aktionen $p = (a_0, \ldots, a_m)$, sodass jede Aktion a_i anwendbar ist auf das Resultat der Aktion a_{i-1} für alle $1 \le i \le m$
- Lösung: Plan, sodass
  - a_0 anwendbar auf Anfangszustand A
  - Jedes Literal von Zielzustand Z kommt auch im Folgezustand von a_m vor

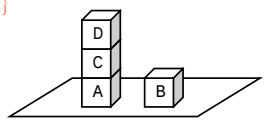
Gliederung

STRIPS-Beispiel: Blockswelt

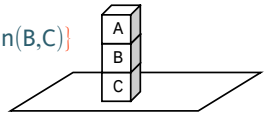
- Blockswelt beinhaltet
  - das Stapeln von Blöcken und das
  - Abbauen von Blockstapeln.
- Logische Sprache
  - $P = \{ontable^1, clear^1, on^2\}$
  - $C = \{A, B, C, D\}$
  - $V = \{x, y\}$

STRIPS-Beispiel: Blockswelt

- Anfangszustand A:
  - A = {on(D,C), on(C,A), clear(D), clear(B), ontable(A), ontable(B)}



- Zielzustand Z:
  - Z = {on(A,B), on(B,C)}



STRIPS-Beispiel: Blockswelt: Aktionsschemata

⇨ACT: put(x, y)

PRE: ontable(x), clear(x), clear(y)

ADD: on(x, y)

DEL: ontable(x), clear(y)

⇨ACT: puttable(x)

PRE: clear(x), on(x, y)

ADD: ontable(x), clear(y)

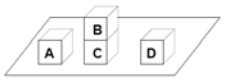
DEL: on(x, y)

⇨ACT: put(x, y)

PRE: on(x, z), clear(x), clear(y)

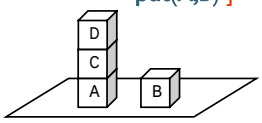
ADD: on(x, y), clear(z)

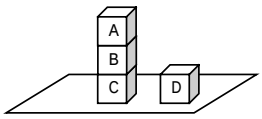
DEL: on(x, z), clear(y)



STRIPS-Beispiel: Blockswelt: Lösung

- **A = {on(D,C), on(C,A), clear(D), clear(B), ontable(A), ontable(B)}**
- **Z = {on(A,B), on(B,C)}**
- **Plan = [puttable(D), puttable(C), put(B,C), put(A,B)]**





STRIPS-Beispiel: Air Cargo Transport

- Air Cargo Transport beinhaltet
  - Beladen von Flugzeugen
  - Transport zwischen Flughäfen
  - Entladen von Flugzeugen
- Logische Sprache
  - P = {Airport¹, Cargo¹, Plane¹, At², In²}
  - C = {C1, C2, P1, P2, JFK, TXL}
  - V = {a, c, p, from, to}

STRIPS-Beispiel: Air Cargo Transport

- Anfangszustand A:
  - A = { At(C1, TXL), At(C2, JFK), At(P1, TXL), At(P2, JFK), Cargo(C1), Cargo(C2), Plane(P1), Plane(P2), Airport(JFK), Airport(TXL) }
- Zielzustand Z:
  - Z = { At(C1, JFK), At(C2,TXL) }

STRIPS-Beispiel: Air Cargo Transport: Aktionen $\mathcal{A}$

- ACT: load**(c, p, a)**
  - **PRE:** At(c, a), At(p, a), Cargo(c), Plane(p), Airport(a)
  - **ADD:** In(c, p)
  - **DEL:** At(c, a)
- ACT: unload**(c, p, a)**
  - **PRE:** In(c, p), At(p, a), Cargo(c), Plane(p), Airport(a)
  - **ADD:** At(c, a)
  - **DEL:** In(c, p)
- ACT: fly**(p, from, to)**
  - **PRE:** At(p, from), Plane(p), Airport(from), Airport(to)
  - **ADD:** At(p, to)
  - **DEL:** At(p, from)

STRIPS-Beispiel: Air Cargo Transport: Lösung

- **A** = { At(C1, TXL), At(C2, JFK), At(P1, TXL), At(P2, JFK), Cargo(C1), Cargo(C2), Plane(P1), Plane(P2), Airport(JFK), Airport(TXL) }
- **Z** = { At(C1, JFK), At(C2,TXL) }
- **Plan** = [
  load(C1, P1, TXL), fly(P1,TXL, JFK), unload(C1, P1, JFK),
  load(C2, P2, JFK), fly(P2, JFK, TXL); unload(C2, P2, TXL)
  ]

Gliederung

Planungsalgorithmen: Planen als Suche im Zustandsraum

- Progression planners
  - Vorwärts gerichtete Suche im Zustandsraum
  - Betrachtet Effekt aller möglichen Aktionen angewendet auf gegebenen Zustand
- Regression planners
  - Rückwärts gerichtete Suche im Zustandsraum
  - Rekonstruiert Vorgänger-Zustände des Zielzustands für gegebene Aktionen

Planungsalgorithmen — Progression Planner

Problemlösen durch Vorwärtssuche:

- Formulierung als Suchproblem
- Beginne mit Anfangszustand A
- Prüfe die Vorbedingungen aller Aktionsschemata
- Berechne Folgezustände durch Ausführen anwendbarer Aktionen

Planungsalgorithmen: Formulierung als Suchproblem — Progression Planner

- Problemraum & Anfangszustand
  - Zustände entsprechen Zuständen des Planungsproblems
  - Fehlende Literale werden als false angenommen
- Zielzustand
  - Jeder Zustand des Planungsproblems, der das Ziel erfüllt
- Aktionen
  - Anwendbar sind alle Aktionen, die Vorbedingungen erfüllen.
  - Füge positive Effekte hinzu, lösche negative Effekte.

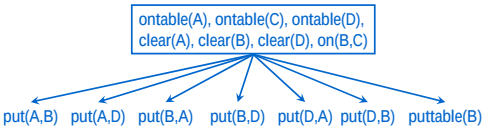
Planungsalgorithmen — Progression Planner

- Das Planungsproblem kann nun als Suchproblem mit beliebigen Suchalgorithmen gelöst werden.
- Problem: Irrelevante Aktionen müssen ausgeführt werden
  - Hoher Verzweigungsgrad
  - Vorwärtssuche ineffizient
  - Verwende Heuristiken

Planungsalgorithmen: Beispiel zur Suche

Progression Planner

- A = { ontable(A), ontable(C), ontable(D), clear(A), clear(B), clear(D), on(B,C) }
- Z = { on(A, B), on(B, C) }



- nur die Aktion put(A, B) ist relevant, alle anderen sind irrelevant

Planungsalgorithmen

Regression Planner

Problemlösen durch Rückwärtssuche:

- Beginne mit Ziel Z_0
- Bestimme alle Aktionen $a_1, \ldots, a_k$ mit $ADD(a_i) \cap Z_0 \neq \varnothing$
- Berechne Vorgängerzustände $Z_{11}, \ldots, Z_{1k}$ durch

$$Z_{1i} = Z_0 - ADD(a_i) \cup PRE(a_i)$$

- Terminiere, wenn Zustand $A' \subseteq A$ erreicht.

Planungsalgorithmen: Bemerkungen

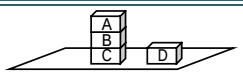
Regression Planner

- Aktion **a** ist relevant für **Z**, wenn $ADD(a) \cap Z \neq \varnothing$.
- Aktion a ist konsistent, wenn es kein Literal in Z löscht.
- Nur relevante und konsistente Aktionen werden angewendet.
- Dieses Planungsproblem ist ebenfalls ein Suchproblem.
- Vorteile:
  - Nur relevante und konsistente Aktionen werden berücksichtigt
  - Häufig geringerer Verzweigungsgrad als bei Vorwärtssuche

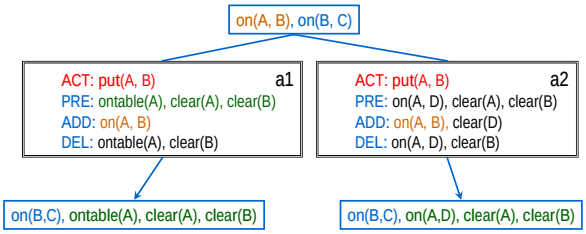
Planungsalgorithmen: Beispiel Rückwärtsplanung

Regression Planner

- Z = { on(A, B), on(B, C) }



- Nur 2 Aktionen relevant und konsistent:



Gliederung

- Einführung
- Klassische Planungssprache – STRIPS
- Beispiele – STRIPS
- Planungsalgorithmen
- Menschliches Problemlösen
- Zusammenfassung

Menschliches Problemlösen

- Beim menschlichen Problemlösen erfolgt die Wahl der *„passenden“* Repräsentation in Wechselwirkung mit dem Problemlöseprozess.
- Fokussierung: Der Mensch konzentriert sich aufgrund der begrenzten Aufmerksamkeitsspanne immer nur auf einen Teil des Problemraums bzw. der Zustandsbeschreibung
- Strategien der Problemraumeinengung bzw. –erweiterung werden verwendet.
  - Dazu müssen oft (perzeptive) Fixierungen auf *„falsche“* Repräsentationen überwunden werden.

Menschliches Problemlösen

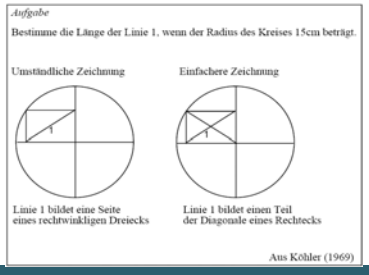
- Beim menschlichen Problemlösen sind meist Lösungen von Teilproblemen bekannt, die wieder verwendet und zu neuen Lösungen komponiert werden
  - Menschen benutzen anstelle einer allgemeinen Suchstrategie häufig Analogien.
  - Übertragung des Wissens bereits gelöster ähnlicher Probleme.
- Menschen verwenden häufig greedy-Strategien

Menschliches Problemlösen

- Der Mensch bildet beim wiederholten Lösen ähnlicher Probleme Automatismen aus (Verhaltensprogramme)
  - Learning by doing (Anderson, 1983)
- These: Ein Charakteristikum menschlicher Intelligenz ist die Fähigkeit zur Wahl problemadäquater Repräsentationen und einer optimalen Problemreduktion.

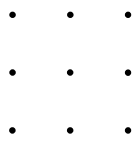
Menschliches Problemlösen

- Die Wahl einer geeigneten Problempräsentation beeinflusst das (einfache) Gelingen der Problemlösung



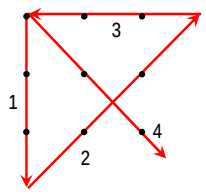
Menschliches Problemlösen: Das 9-Punkte Problem

- Neun in Form eines Quadrates angeordnete Punkte sollen durch vier gerade Linen in einem Zug verbunden werden.



Menschliches Problemlösen: Das 9-Punkte Problem

- Neun in Form eines Quadrates angeordnete Punkte sollen durch vier gerade Linen in einem Zug verbunden werden.



Menschliches Problemlösen: Das 9-Punkte Problem

- Die sprachliche Formulierung bzw. perzeptive Fixierung lenkt die Aufmerksamkeit auf eine unvollständige Repräsentation des Problemraumes.
- Die notwendige Suchraumerweiterung kann nur über eine abstrakte Ordnungsregel, durch Weiterführung des Aufbaugesetzes der Konfiguration erfolgen.

Gliederung

- Einführung
- Klassische Planungssprache – STRIPS
- Beispiele – STRIPS
- Planungsalgorithmen
- Menschliches Problemlösen
- Zusammenfassung

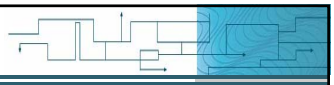
Zusammenfassung: Planen kombiniert Logik und Suche

- Planungssysteme sind Problemlösealgorithmen, die auf einer expliziten propositionalen Repräsentation der Zustände und Aktionen operieren.
- Die Planungssprache STRIPS beschreibt
  - Aktionen durch Vorbedingung und Effekt (ADD/DEL)
  - Zustände durch Konjunktionen von Literalen
  - Action Description Language ist eine Erweiterung von STRIPS

Zusammenfassung

- Suche im Zustandsraum kann vorwärtsgerichtet (progressiv) oder rückwärtsgerichtet (regressiv) verlaufen
  - Die Wahl des richtigen Verfahrens ist eine Kunst
- „Planungsalgorithmus sollte logische Struktur des Problems ausnutzen“
  - Informationen zu nicht klassischen Planungssystemen im Anhang, in den Referenzen und bei Russel & Norvig, 2003
- Exkurs in menschliches Problemlösen als Abschluss des VL-Blocks Problemlösen.

DAI-Labor
TU Berlin



Informierte Suchverfahren

nächster Termin: 17.11.2005

Logik & Wissensrepräsentation 1

Brijnesh J Jain, Stefan Fricke
bjj@dai-labor.de stefan.fricke@dai-labor.de

A|O|T
Agententechnologien in betrieblichen Anwendungen und der Telekommunikation

Action Description Language als Erweiterung von STRIPS

- Positive und negative Literale in Zuständen
- Open World Assumption: Nicht erwähntes ist unbekannt
- Effekt P $\wedge \neg$ Q ==> **add P** und $\neg$ **Q, del** $\neg$P und **Q**
- Quantifizierte Variablen in Zielen
- Auch Disjunktionen in Zielen erlaubt
- Bedingte Effekte
- Typisierte Variablen
- Gleichheitsprädikat

Gliederung

- Einführung
- Klassische Planungssprache – STRIPS
- Beispiele – STRIPS
- Planungsalgorithmen
- Planungssysteme
- Menschliches Problemlösen
- Zusammenfassung

Planungssysteme

- Hierarchisches Planen
  - Erzeugung von Plänen auf verschiedenen Abstraktionsebenen
  - Klassischer Ansatz: NOAH (Sacerdotti, 1974)
- Deduktives Planen
  - Beweis von Planspezifikationen durch Resolution, basierend auf Situationskalkül (Green, 1969)
  - Planen als deduktive Programmsynthese (Manna & Waldinger 1986)
  - Modallogische Verfahren (dynamische Logiken, Temporallogik)

Planungssysteme

- Nicht-klassisches Planen
  - Behandlung von Unsicherheit
  - Beispiel: Universelles Planen (Schoppers, 1987)
    - Beschreibung des Handlungswissen über gesamten Bereich
    - Robust gegenüber unvorhergesehene Änderungen
    - Effizient
- Weitere wichtige Ansätze (s. Russel & Norvig, 2003)
  - Anytime Planner
  - Planen und Lernen
  - Partial Order Planning
  - Graphplan

Planungsalgorithmen