Planen Grundlagen

Dr. Christian Meilicke
Research Group Data and Web Science
Universität Mannheim

Teile der Vorlesung basieren auf einem
Foliensatz von Prof. Dr. Heiner Stuckenschmidt

Wo sind wir

- Einleitung: Was ist KI, Agenten, Umwelteigenschaften

- Suche mit Suchbaum
- Lokale Suche
- Constraints
- Logik
- Planen
  - als Suche
  - ~~mit Constraints~~
  - mit Logik
  - ~~mit Planungsgraph~~

Planen - Überblick

- Einführung
  - Was ist Planen
  - Definition: Planungsproblem
  - Erste Beispiele
- Notation und Ausdrucksstärke
  - Handlungsschemata
  - Planungssprachen
  - Modellierung mit PDDL
- Beispiele
  - Tourenplanung
  - Towers of Hanoi
- Planen als Suche

Was ist Planen ?

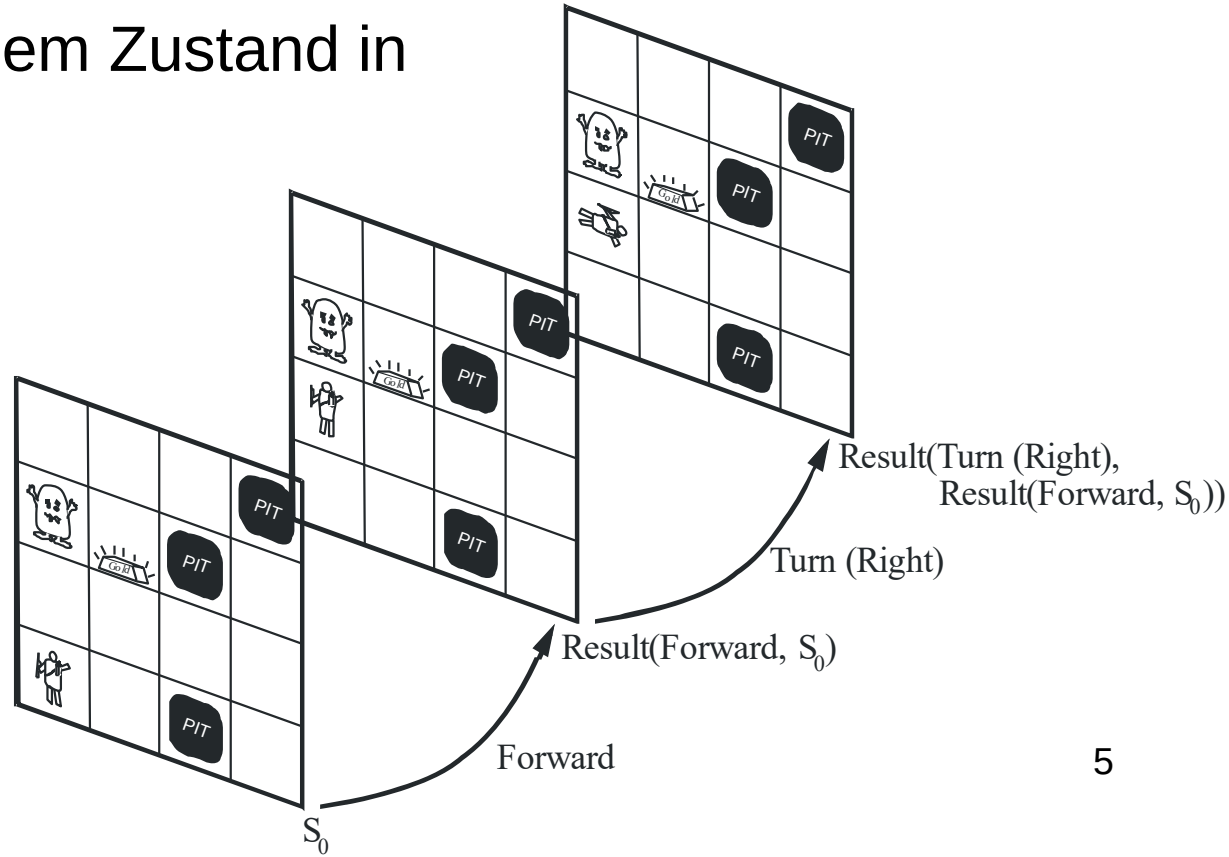
- Generierung einer Handlungssequenz, die von einem gegebenen Zustand zu einem bestimmten Ziel führt

- Beispiele:
  - Handlungsplanung (Roboter, NPCs in Computerspielen, Solitärspiele)
  - Ablaufplanung (Projektplanung, Fertigungsplanung, Präsentationsgenerierung)

- Hier: Einschränkung auf „klassisches Planen“
  - Fully observable, deterministic, discrete

Planen vs. Logik

- Logisches Schließen betrifft einen bestimmten Zustand
- Planung beinhaltet zusätzlich den Übergang von einem Zustand in einen anderen



Planen vs. Suche

- Probleme generischer Suchverfahren
  - Vielzahl irrelevanter Handlungen
  - Keine problemunabhängigen Heuristiken
  - Keine Möglichkeit, Teilziele zu berücksichtigen

- Planen als Problemtyp
  - Spezielle Planungssprachen zur Definition von Zielen und Handlungen
  - Spezielle Datenstrukturen und Heuristiken zur Darstellung des Problemraums

Frame-Problem

- Problem:
  - Handlungen beschreiben was sich ändert, nicht was gleich bleibt (z.B. *unknown*)
  - Alle Aspekte aufzuzählen, die gleich bleiben ist sehr aufwendig

- Lösungen:
  - Spezielle Logiken, die zwischen veränderlichen und unveränderlichen Aspekten unterscheiden
  - Explizite Annahme im Planungsalgorithmus: Alles was nicht explizit negiert wird gilt weiterhin

Frame-Problem: Beispiel

- Robinson Roulette ist ein typisches Planungsproblem
  - Letztes Jahr haben wir dieses mit Logik gelöst
  - Frame-Axiome mußten explizit eingeführt werden

```
// effect 2: well not really an effect, just do not
// allow to remove something from the outer circle
for (int t = 0; t < n-1; t++) {
    for (int pos = 0; pos < n; pos++) {
        Clause c = new Clause();
        c.addLiteral(new Literal("b_" +  pos + "," + t, false));
        c.addLiteral(new Literal("b_" +  pos + "," + (t+1), true));
        clauses.add(c);
    }
}
```

z.B. $b_{3,5} \rightarrow b_{3,6}$

- Kann in speziellen “Planungslogiken” (= Planungsprachen) implizit behandelt werden

Definition Planungsproblem

- P = (A,O,I,G)
- Zustände A
  - Menge von Aussagen, die verwendet werden um O, I, G zu beschreiben
- Anfangs- oder Initialzustand $I \subseteq A$
  - Vollständige Beschreibung eines Zustandes
- Ziele und Teilziele $G \subseteq A$
  - Menge von Aussagen, die gelten sollen, damit das Ziel erreicht ist
  - Unvollständige Beschreibung eines Zustands
- Handlungen O
  - Bezeichnung der Handlung
  - Bedingungen, die gelten müssen, damit die Handlung möglich ist
  - Effekte, die eintreten, wenn die Handlung ausgeführt wird

Einfaches Beispiel

- Initialzustand
  - Vollständige Beschreibung wahrer Aussagen als Menge logischer Formeln (**Closed-World**)
  - z.B.: *unknown*
- Ziele und Teilziele
  - Partielle Beschreibung des gewünschten Zustands als Formel
  - *Z.B.: rich* $\wedge$ *famous*
- Handlungen (z.B.: *work-extremely-hard*)
  - Voraussetzungen (z.B. *good-job, smart*)
  - Konsequenzen: (z.B. *rich* $\wedge$ $\neg$*happy*)
- Lösung:
  - Sequenz von Handlungen, die tatsächlich ausgeführt werden können, und deren Konsequenz die Zielformeln enthält

Komplexes Beispiel

- Tourenplanung
  - Zustände
    - $At(p_i,a_j)$
      Flugzeug p_i ist am Flughafen a_j
  - Ziel
    - At(p,a), Plane(p), Airport(a) $\wedge$ ...
  - Handlungen
    - Fly(p,from,to)
      - Bedingung: At(p,from)
      - Konsequenz: At(p,to), $\neg$ At(p,from)

- Statt Aussagen Verwendung von Aussagenschemata
  - Ähnelt in der Schreibweise Prädikatenlogik
  - Aussagenlogik, da endlich viele Entitäten

Repräsentationsprobleme

- Zeitliche Aspekte
  - At(p,from) $\wedge$ Fly(p,from,to) $\rightarrow \neg$ At(p,from)
  - Aussage und Ihre Negation sollen wahr sein
  - Unterscheidung von Zeitpunkten notwendig
    - At(p,from)^0 vs. At(p,from)^1
- Vielzahl möglicher Handlungen
  - Fly Aktionen für jede Kombination von p, from und to
- Problemgrösse explodiert!
  - |Zeitpunkte| $\times$ |Planes| $\times$ |Airports|^2

Planen I - Überblick

- Einführung
  - Besonderheiten beim Planen
  - Definition: Planungsproblem
  - Erste Beispiele
- Notation und Ausdrucksstärke
  - Handlungsschemata
  - Planungssprachen
  - Modellierung mit PDDL
- Beispiele
  - Tourenplanung
  - Towers of Hanoi
- Planen als Suche

Handlungen / Zustandsübergange

- Für jede Handlung muss spezifiziert werden
  - Unter welchen Bedingungen kann man die Handlung ausführen
  - Was verändert sich (und was bleibt gleich) wenn man die Handlung ausführt

- Problem:
  - Sehr viele konkrete Handlungen
- Idee:
  - Statt konkrete Handlungen abstrakte Handlungsschemata angeben

Handlungschemata

- Prädikat, das die Handlung und mögliche Parameter beschreibt
  - Fly(p,from,to)
- Formel, die Bedingungen beschreiben:
  - At(p, from) $\wedge$ Fueled(p)
- Formel, die positive und negative Effekte beschreiben
  - $\neg$At(p, from) $\wedge$ At(p, to)
  - Effekte werden auch of als Listen Add und Del dargestellt

Annahme: Alle Aussagen, die nicht explizit in den Effekten erwähnt werden, bleiben wie bisher!

Beispiel

Action (**Load** (c, p, a),
PRECOND: At (c, a) ∧ At (p, a) ∧ Cargo (c) ∧ Plane (p) ∧ airport (a)
EFFECT: ¬At (c, a) ∧ In (c, p))

Action (**Unload** (c, p, a),
PRECOND: In (c, p) ∧ At (p, a) ∧ Cargo (c) ∧ Plane (p) ∧ Airport (a)
EFFECT: At (c, a) ∧ ¬In (c, p))

Action (**Fly** (p, from, to),
PRECOND: At (p, from) ∧ Plane (p) ∧ Airport (from) ∧ Airport (to)
EFFECT: ¬At (p, from) ∧ At (p, to))

Initial State:
Init (At (C1, SFO) ∧ At (C2, JFK) ∧ At (P1, SFO) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

Goal State:
Goal (At (C1, JFK) ∧ At (C2, SFO))

Planungsschritte 1+2

Init (At (C1, SFO) ∧ At (C2, JFK) ∧ At (P1, SFO) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

1)

Action (**Load** (C1, P1, SFO),
PRECOND: At (C1, SFO) ∧ At (P1, SFO) ∧ Cargo (C1) ∧ Plane (P1) ∧ airport (SFO)
EFFECT: ¬At (C1, SFO) ∧ In (C1, P1))

S1 (In (C1, P1) ∧ At (C2, JFK) ∧ At (P1, SFO) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

2)

Action (**Fly** (P1, SFO, JFK),
PRECOND: At (P1, SFO) ∧ Plane (P1) ∧ Airport (SFO) ∧ Airport (JFK)
EFFECT: ¬At (P1, SFO) ∧ At (P1, JFK))

S2 (In (C1, P1) ∧ At (C2, JFK) ∧ At (P1, JFK) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

Planungsschritte 3+4

S2 (In (C1, P1) ∧ At (C2, JFK) ∧ At (P1, JFK) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

---

3) Action (**Load** (C2, P2, JFK),
PRECOND: At (C2, JFK) ∧ At (P2, JFK) ∧ Cargo (C2) ∧ Plane (P2) ∧ airport (JFK)
EFFECT: ¬At (C2, JFK) ∧ In (C2, P2))

S3 (In (C1, P1) ∧ In (C2, P2) ∧ At (P1, JFK) ∧ At (P2, JFK) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

4) Action (**Fly** (P2, JFK, SFO),
PRECOND: At (P2, JFK) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO)
EFFECT: ¬At (P2, JFK) ∧ At (P2, SFO))

S4 (In (C1, P1) ∧ In (C2, P2) ∧ At (P1, JFK) ∧ At (P2, SFO) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

Planungsschritte 5+6

S4 (In (C1, P1) ∧ In (C2, P2) ∧ At (P1, JFK) ∧ At (P2, SFO) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

5)

Action (**Unload** (C1, P1, JFK),
PRECOND: In (C1, P1) ∧ At (P1, JFK) ∧ Cargo (C1) ∧ Plane (P1) ∧ Airport (JFK)
EFFECT: At (C1, JFK) ∧ ¬In (C1, P1))

S5 (At (C1, JFK) ∧ In (C2, P2) ∧ At (P1, JFK) ∧ At (P2, SFO) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

6)

Action (**Unload** (C2, P2, SFO),
PRECOND: In (C2, P2) ∧ At (P2, SFO) ∧ Cargo (C2) ∧ Plane (P2) ∧ Airport (SFO)
EFFECT: At (C2, SFO) ∧ ¬In (C2, P2))

S6 (At (C1, JFK) ∧ At (C2, SFO) ∧ At (P1, JFK) ∧ At (P2, SFO) ∧ Cargo (C1) ∧ Cargo (C2) ∧ Plane (P1) ∧ Plane (P2) ∧ Airport (JFK) ∧ Airport (SFO))

Planungssprachen

- Logik als Basis der Problembeschreibung

	STRIPS(1972)	ADL (1989)
Zustände	Nur Aussagen ohne Negation: poor $\wedge$ unknown	Auch negierte Aussagen, Gleichheit und Typisierung: poor $\wedge$ $\neg$known, (heiner = hs), hs:Person
	Closed World Assumption: Unbekannte Aussagen sind falsch	Open World Assumption: Kein wissen über unbekannte Aussagen
Ziele/ Vorbedingungen	Konjunktionen von Aussagen: rich $\wedge$ famous	Volle Aussagenlogik $\neg$poor $\wedge$ (famous $\vee$ smart)
	Nur einfache Aussagen: rich $\wedge$ famous	Quantifizierte Variablen möglich: $\exists$x.married-to(hs,x)$\wedge$beautiful(x)
Effekte	Effekte sind Konjunktionen: P $\wedge$ $\neg$Q	Bedingte Effekte möglich: C $\rightarrow$ P $\wedge$ $\neg$Q
	P $\wedge$ $\neg$Q heisst P gilt jetzt und Q gilt nicht mehr	P $\wedge$ $\neg$Q heisst P und $\neg$Q gelten jetzt und $\neg$P und Q gelten nicht mehr

PDDL: Planning Domain Definition Language

- Standardsprache zur Beschreibung von Planungsproblemen
  - PDDL 1.0: ähnlich ADL
  - PDDL 2.1:
    - numerische Variablen
    - Temporale Aspekte
  - PDDL 2.2:
    - Abgeleitete Prädikate in Zuständen
    - Planen mit Zeitpunkten

Planen I - Überblick

- Einführung
  - Besonderheiten beim Planen
  - Definition: Planungsproblem
  - Erste Beispiele
- Notation und Ausdrucksstärke
  - Handlungsschemata
  - Planungssprachen
  - Modellierung mit
- Beispiele
  - Tourenplanung
  - Towers of Hanoi
- Planen als Suche

Beispiel: Türme von Hanoi

- Planungssysteme erwarten zwei Files als Eingabe
  - Allgemeine Domänenbeschreibung
    - Handlungen mit Vorraussetzungen und Effekten
  - Faktenfile
    - Liste der Entitäten
    - Startzustand
    - Zielzustand
- Faktenfile austauschbar

Domänenbeschreibung

```
(define (domain hanoi)
   (:predicates    (on ?disk1 ?disk2)
         (smaller ?disk1 ?disk2)
         (clear ?disk))
  (:action MOVE
      :parameters   (?disk ?source ?dest)
      :precondition
        (and    (clear ?disk)
           (on ?disk ?source)
           (clear ?dest)
           (smaller ?disk ?dest))
      :effect
        (and    (on ?disk ?dest)
           (not (on ?disk ?source))
           (not (clear ?dest))
           (clear ?source))))
```

Fakten

```
(define (problem hanoi4)
  (:domain hanoi)
  (:length (:parallel 15))
  (:objects D1 D2 D3 D4 P1 P2 P3)
  (:init
    (on D1 D2)
    (on D2 D3)
    (on D3 D4)
    (on D4 P1)
    (clear D1)
    (clear P2)
    (clear P3)
    weiter rechts
    (smaller D1 D2)
    (smaller D1 D3)
    (smaller D1 D4)
    (smaller D1 P1)
    ...
  )

  (:goal
    (and
      (on D1 D2)
      (on D2 D3)
      (on D3 D4)
      (on D4 P3)
    )
  )
)
```

Flughafen Beispiel in PDDL

Aktionen:

```
(define (domain air-cargo)

        (:requirements :typing :adl)
        (:types cargo plane airport)
        (:predicates (at ?t - (either cargo plane) ?a - airport)
        (in ?c - cargo ?p - plane))

        (:action load
        :parameters (?c - cargo ?p - plane ?a - airport)
        :precondition (and (at ?c ?a) (at ?p ?a))
        :effect (and (not (at ?c ?a)) (in ?c ?p)))

        (:action unload
        :parameters (?c -cargo ?p - plane ?a - airport)
        :precondition (and (in ?c ?p) (at ?p ?a))
        :effect (and (at ?c ?a) (not (in ?c ?p))))

        (:action fly
        :parameters (?p - plane ?a1 ?a2 - airport)
        :precondition (and (at ?p ?a1) (not (= ?a1 ?a2)))
        :effect (and (not (at ?p ?a1)) (at ?p ?a2)))
)
```

Anfangs- und Zielzustand:

```
(define (problem sfo-jfk)
        (:domain air-cargo)
        (:objects c1 c2 - cargo sfo jfk - airport p1 p2 - plane)
        (:init (at c1 sfo)
        (at p1 sfo)
        (at c2 jfk)
        (at p2 jfk)

         (:goal (and (at c1 jfk) (at c2 sfo)))
)
```

Demo

- Wie man einen Planer benutzt
  - http://editor.planning.domains/

Planen I - Überblick

- Einführung
  - Besonderheiten beim Planen
  - Definition: Planungsproblem
  - Erste Beispiele
- Notation und Ausdrucksstärke
  - Handlungsschemata
  - Planungssprachen
  - Modellierung mit PDLL
- Beispiele
  - Tourenplanung
  - Towers of Hanoi
- Progressives Planen als Suche

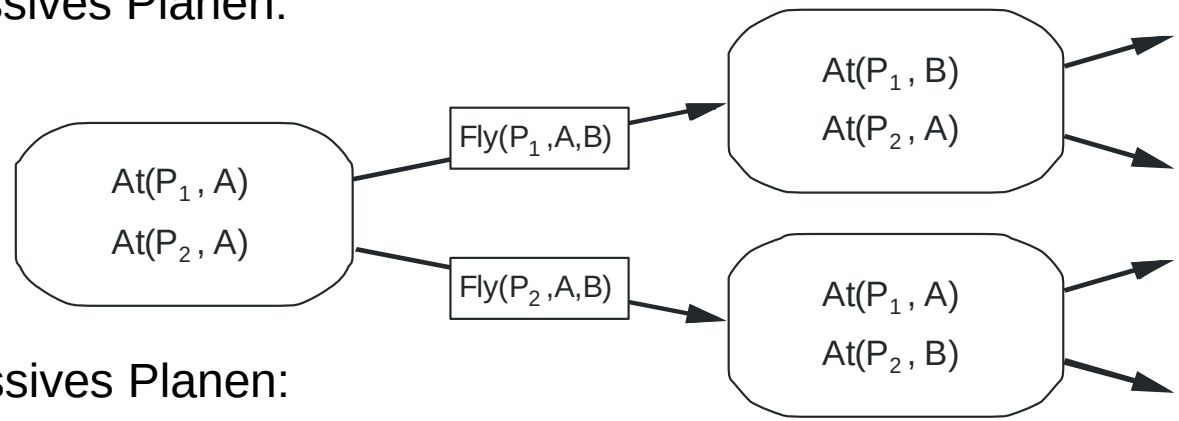
STRIPS Planen als Suche

- Anfangszustand:
  - gegebener Anfangszustand des Planungsprobelms
- Aktionen:
  - Handlungsschemata, deren Vorbedingungen erfüllt sind
  - Folgezustand ergibt sich durch Löschen von negierten und Hinzufügen von nicht-negierten Effekten
- Zielzustand:
  - Zustand, in dem die Zielbedingungen des Planungsproblems erfüllt sind.
- Vollständige Suchverfahren wie A* können verwendet werden, um Planungsprobleme zu lösen
- Heuristiken können automatisch aus der Repräsentation abgeleitet werden !

Planungsstrategien

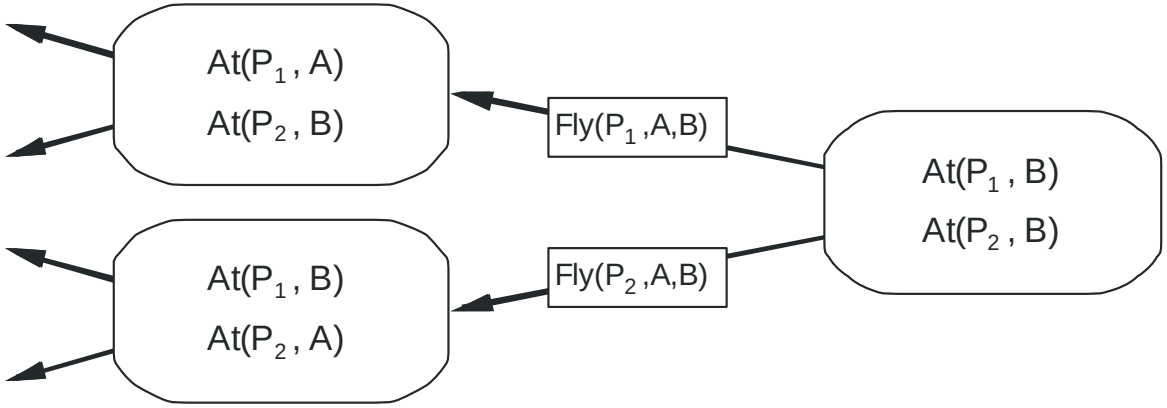
Progressives Planen:

(a)



Regressives Planen:

(b)



Progressives Planen

- Gegeben ist ein Planungsproblem (A,O,I,G)
  - Der Suchraum ist die Potenzmenge von A
  - Startzustand s_0 ist I
  - Zielzustände sind alle Zustände s so dass $G \subseteq s$
  - Mögliche Handlungen in einem Zustand s sind alle op aus O so dass $Prec(op) \subseteq s$
  - Der neue Zustand nach Anwendung des Operators op ist $s - Del(op) \cup Add(op)$
  - Die Kosten sind für jede Handlung gleich 1

Suchverfahren

- Variante der A* Suche
  - $f(n) = g(n) + W \cdot h(n)$
  - (Entspricht A* für W=1)

- $g(n) = G \cap n$

- Je grösser W, desto ähnlicher ist die Methode der Greedy Suche
  - Lösungen werden schneller gefunden
  - Qualität der Lösungen nimmt ab

- Keine optimalen, aber oft gute Lösungen für grosse W

Heuristiken

- Wie wurden noch mal Heuristiken entwickelt?
- Was heißt nochmal „admissible“ (zulässig)?
- Vorschläge für Planungs-Heuristiken?

Heuristiken I

- Idee: Vereinfachung der Vor- bzw. Nachbedingungen:
  - Leere Vorbedingungen
  - Keine negativen Effekte
- Sind diese Heuristiken admissible ?

Heuristiken II

- Idee: Berechnung der Schritte, die nötig sind um einzelne Aussagen des Ziels zu erreichen
  - Berechnung eines kürzesten Weges vom aktuellen Zustand in einen Zustand, in dem die Aussage gilt
  - Entspricht kürzester Wege Suche in Graphen
  - polynomielle Komplexität

- Geeignete Kombination der Heuristiken für die Gesamtmenge von Aussagen:
  - Additive Heuristik: h ist Summe der Einzelkosten
  - Maximum Heuristik: h ist das Maximum der Einzelkosten

- Sind diese Heuristiken admissible ?

Zusammenfassung und Ausblick

- Einführung in Planen
  - Definition
  - Spezielle „Datenstruktur“
    - Ermöglicht komfortable Problembeschreibung
    - Ableitung problemunabhängiger Heuristiken
  - PDDL und online Solver

- Wer mehr wissen will:
  - http://projects.laas.fr/planning/book.pdf
    Aber Achtung: Deutlich komplizierterer Formalismus
  - Erweiterungen
    - Berücksichtigung von Zeit
    - Verfeinerung von Aktivitäten/Plänen
    - Nicht-deterministisches Ausführen von Handlungen

Ende

- Was war schlecht:
  - Noch immer können wir keine KI entwicklen, welche die Menscheit einmal vernichten wird
  - Aber egal: das schafft die Menschheit auch ohne unsere Hilfe

- Was war gut:
  - Interessante Algorithmen
  - Grundprinzip:
    - Schätze die Komplexität des Problems ab
    - Überführe das Problem in ein Suchproblem
      - Mache die Suche durch Heuristiken und „andere Kniffe“ effizient
      - Oder verwende einen Formalismus in dem das automatisch passiert

Klausur

Am 22.12. kurz vor Weihnachten

Fragen => ILIAS