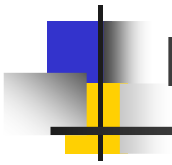


Konditionale Effekte in Graphplan

Robert Feldmann

Oktober 2002

Gliederung

- 1. Einführung in das Planen
- 2. STRIPS
- 3. Graphplan
- 4. Erweiterungen von Graphplan
- 5. Zusammenfassung
- Quellen / Literatur

Einführung in das Planen

- Agenten agieren in einer Welt um bestimmte Ziele zu erreichen →Algorithmus für das Agieren wird benötigt
- 2 Ansätze: Planen, situationsbezogenes Handeln
- Planen, wenn:
  - Aktionen in geordneter Reihenfolge
  - Vorausschauendes Handeln nötig
  - Bsp.: Reise nach Afrika (Hotel,Flug, Taxi buchen, Visum ...)
- Situationsbezogens Handeln
  - Geeignete Handlung kann leicht aus dem aktuellen Zustand abgeleitet werden
  - Bsp.: Flipper spielen

Einführung in das Planen

- Kriterien für Planer
  - Refinement/Verfeinerung: von Plänen durch Hinzufügen von Aktionen und Bedingungen
  - Retraction: umgekehrt – Löschen von Aktionen
  - Transformational: Mischform obiger Formen
  - Generative Planer vs. Case-based Planer
- Jeder Planungsalgortihmus besitzt folgende Eingaben
  1. Eine Beschreibung der Welt in einer gegebenen Sprache
  2. Eine Zielbeschreibung des Agenten
  3. Eine Beschreibung möglicher Aktionen → „domain theory"
- Die Ausgabe ist eine Liste von Aktionen, die in der festgelegten Reihenfolge ausgeführt in jeder Welt, die den Anfangsbedingungen genügt, das Ziel erreichen, d.h. ein Plan

Einführung in das Planen

- Welche Sprache verwendet wird, hängt von der Problemklasse ab
  - zB. Aussagenlogik, P1, STRIPS, PDDL,...
- Häufig verwenden Planer folgende Annahmen
  - Atomizität der Zeit, d.h. nur pseudoparallele Aktionen
  - Deterministische Effekte
  - Omniscience/Allwissenheit des Agenten
  - Veränderung nur durch Agenten, d.h. statische Umwelt, keine anderen Agenten

  → Unrealistisch, aber häufig erlauben diese Annahmen erst effiziente Planungsalgorithmen

STRIPS

- STRIPS (STanford Research Institute Problem Solver)
  - Vorgestellt 1970
  - Steuerte einen mobilen Roboter „Shakey"
- STRIPS-Repräsentation weit verbreitet
- STRIPS-Operatoren bestehen aus
  - Name
  - Liste typisierter Parameter
  - Vorbedingungen/preconditions: Konjunktion von pos. Literalen
  - Nachbedingungen/effects: Konjunktion von pos. (add-Liste) und neg. (del-Liste) Literalen
- STRIPS-Aktionen, sind Grundinstanzen von Operatoren
  - Spezielle Aktionen no-op(a) für jedes pos. Literal a

STRIPS

- STRIPS-Zustände
  - Menge von Grundatomen (Bedingungen)
  - Keine neg. Literale, da aufgrund von CWA (closed world assumption) bei fehlenden pos. Literalen automatisch die neg. Literale als wahr angenommen werden
- STRIPS-Problem
  - Menge von STRIPS-Operatoren
  - Menge von (typisierten) Objekten
  - Anfangszustand
  - Menge von Zielbedingungen
  - Gesucht wird ein Plan

STRIPS

- STRIPS-Plan P: Menge von Aktionen mit Aufrufzeitpunkten, wobei
  - Aktionen nur bei erfüllten Vorbedingungen ausgeführt werden
  - Aktionen sich nicht gegenseitig behindern
  - P alle Zielbedingungen wahr macht
- Beachte:
  - Allg. kann ein Ziel auch ein Zeitintervall umfassen, jedoch in STRIPS sind Ziele auf den Zustand direkt nach Abschluss des Planes bezogen
  - Zielzustände sind positive Literale
  - Rechnen nur mit positiven Literalen → CWA

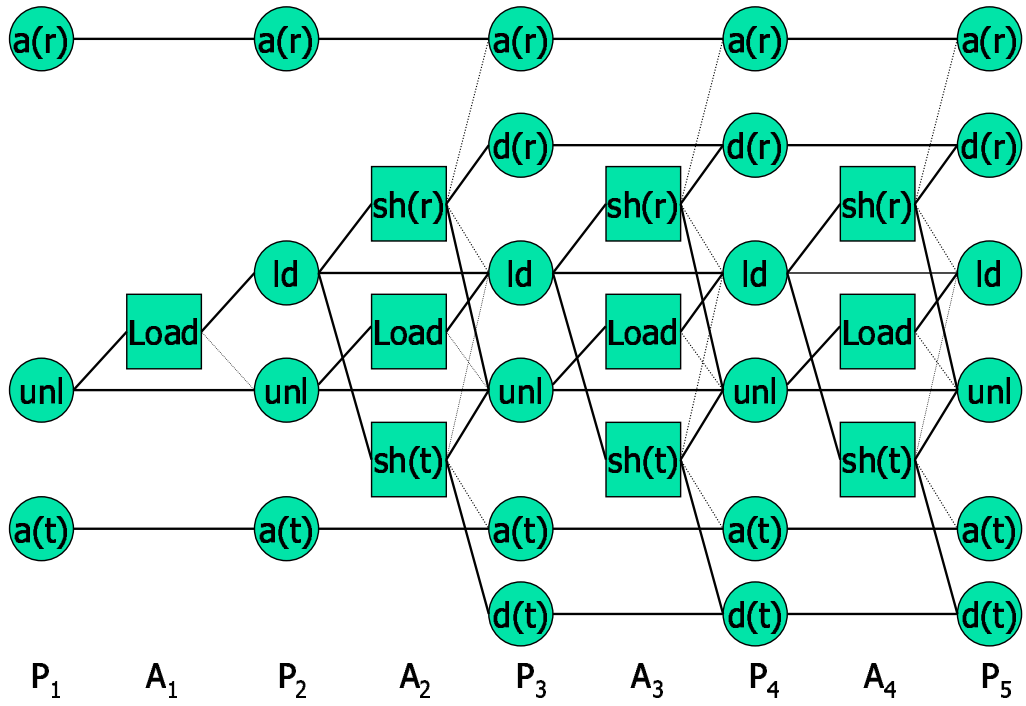
Graphplan

- Veröffentlicht 1995 Blum & Furst
- wahrscheinlich der momentan schnellste STRIPS-Planer
- Ziel: Erweiterung des Sprachumfangens mit dem Graphplan umgehen kann
- Werden uns deshalb später folgenden Problemen zuwenden
  - Negative und Disjunktive Vorbedingungen
  - Konditionale Effekte
- Doch zunächst: Graphplan verwendet Plangraphen
- Was ist ein Plangraph?

Graphplan

- Gerichteter Level-Graph mit
  - 2 Arten von Knoten: Aktionen und Propositionen(Bedingungen)
  - 3 Arten von Kanten: add, delete, precondition
- Beginn bei Propositionslevel P_1
- Auf P_i folgt A_i – auf A_i folgt P_{i+1}
- Aktion A kommt in A_i vor
  - falls all ihre Vorbedingungen in P_i vorkommen und sich nicht ausschliessen
  - **dann** kommen alle Effekte von A in P_{i+1} vor
- Graphplan definiert eine Ausschlussrelation (mutex) und verkleinert so den Suchraum beim Backchaining.
- Dazu später mehr, jetzt ein Beispiel eines Plangraphen

Graphplan



Graphplan

- Ausschlussrelation – mutex (mutual exclusion)
- Für Aktionen
  - Interferenz/Inkonsistenz:
    - A_1 und A_2 schliessen sich aus, wenn A_1 eine Vorbedingung oder einen add-Effekt von A_2 löscht (oder umgekehrt)
  - Inkompatible Vorbedingungen
    - A_1 und A_2 schliessen sich aus, wenn sich mindestens 2 ihrer Vorbedingungen ausschliessen
- Für Propositionen
  - Zwei Propositionen p und q schliessen sich aus im Level P_i, wenn alle Wege p zu erreichen und alle Wege q zu erreichen sich gegenseitig ausschliessen, d.h. wenn alle Aktionen des Levels A_{i-1} die p erzeugen und alle Aktionen des Levels A_{i-1} die q erzeugen sich paarweise gegenseitig ausschliessen.

Graphplan

- Wir können nun den Algorithmus definieren (siehe [Brewka])

i := 1;
Prop1 := Anfangszustand;
while kein Plan gefunden [und letzter Level nicht identisch mit vorletztem] **do**
begin
erweitere Plangraph um Akti und Propi+1 ;
prüfte, ob es gültigen Plan der Länge i gibt;
i := i + 1
end;
if Plan gefunden **then** gib Plan aus **else** ‚kein Plan existiert'

Graphplan

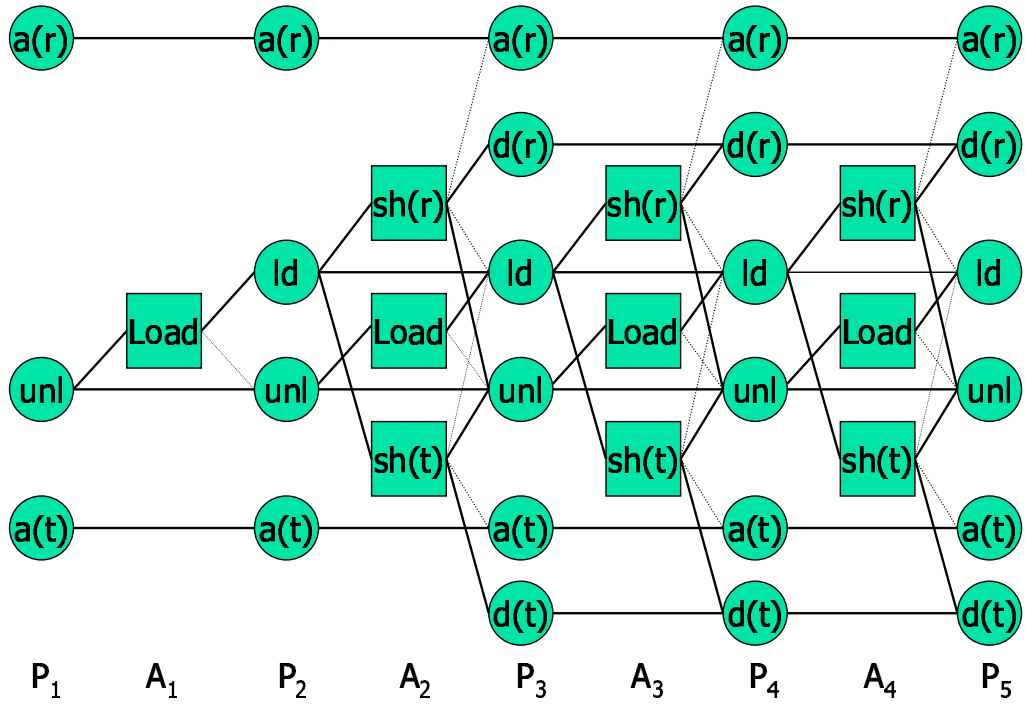
- 1. Erweiterung um Aktions-Level A_i:
  a) Füge Aktion A in A_i ein, wenn alle Vorbedingungen von A in P_i enthalten und nicht als sich ausschließend markiert sind.
  b) Füge alle no-op Aktionen und precondition links ein.
  c) generiere für jede Aktion A in A_i Liste aller Aktionen, die A ausschließt.
- 2. Erweiterung um Propositions-Level P_i+1:
  a) Füge alle Atome aus Nachbedingungen der Aktionen in A_i ein, zusammen mit entsprechenden Add- und Delete-Links.
  b) Markiere p, q in P_i+1 als sich ausschließend, wenn alle Aktionen, die p generieren, und alle, die q generieren, sich ausschließen.
- 3. Plansuche

  Gegeben Menge G von Zielen in t, suche sich nicht ausschließende Menge von Aktionen in t-1, die alle Ziele in G wahrmachen. Deren Vorbedingungen sind Ziele in t-1, usw.

Graphplan

- Nochmal das Shooting-Beispiel
- shoot(x: animal):
  - pre: loaded
  - add: dead(x), unloaded
  - del: alive(x), loaded
- load:
  - pre: unloaded
  - add: loaded
  - del: unloaded
- init: alive(r), alive(t), unloaded
- goal: dead(r), dead(t)

Graphplan



Erweiterungen von Graphplan

- Ziel Sprachumfang erweitern
- Relativ einfach ist die Erweiterung um negative und disjunkte Vorbedingungen
- Negative Vorbedingungen:
  - Del-Kante wird ersetzt durch add-Kante des negativen Literales
  - Zwei Propositionen p und ~p sind auf allen Levels mutex
- Disjunktive Vorbedingungen
  - Umwandeln der Vorbedingung in DNF (disjunktive Normalform)
  - Während der Expansionsphase wird eine Aktion mit mehreren Disjunkten in der Vorbedingung genau dann dem Graphen hinzugefügt, wenn all Konjunkte mindestens eines Disjunktes im vorhergehenden Level vorkommen und nicht mutex sind.

Erweiterungen von Graphplan

- Bsp.:
  - Take-sick-bag:
    - Pre: (be-sick AND be-on-plane) OR (be-sick AND be-in-car)
    - Eff: have-sick-bag, ~have-empty-hands
- Während der Backchaining phase: Wenn der Planer im Level A_i Aktion A betrachtet, muss er alle möglichen Disjunkte der Verbindung als mögliche Unterziele annehmen (Vollständigkeit).

  →starke Erhöhung der Backchaining Suchzeit

  →möglichst wenig Disjunktionen verwenden

Erweiterungen von Graphplan

- Was sind konditionale Effekte?
  - Eine Aktion kann abhängig von der Situation unterschiedliche Effekte haben
  - When-Klause: (when antecedent consequent)
  - Falls Antecedent bei t=i gilt, so gilt consequent bei t=i+1
- Bsp.: „Move-Briefcase-domain"

```
movebriefcase (?loc ?new)
    :prec (and (at briefcase ?loc) (location ?new) (not (= ?loc ?new)))
    :effect (and (at briefcase ?new) (not (at briefcase ?loc))
                 (when (in paycheck briefcase)
                       (and (at paycheck ?new)  (not (at paycheck ?loc))))
                 (when (in keys briefcase)
                       (and (at keys ?new) (not (at keys ?loc)))))
```

Erweiterungen von Graphplan

- Der „Full Expansion" Ansatz(1)

```
movebriefcaseempty (?loc ?new)
    :prec (and (at briefcase ?loc) (location ?new) (not (= ?loc ?new))
               (not (in paycheck briefcase)) (not (in keys briefcase)))
    :effect (and (at briefcase ?new) (not (at briefcase ?loc)))

movebriefcasepaycheck (?loc ?new)
    :prec (and (at briefcase ?loc) (location ?new) (not (= ?loc ?new))
               (in paycheck briefcase) (not (in keys briefcase)))
    :effect (and (at briefcase ?new) (not (at briefcase ?loc))
                 (at paycheck ?new) (not (at paycheck ?loc)))
```

Erweiterungen von Graphplan

- Der „Full Expansion" Ansatz (2)

```
movebriefcasekeys (?loc ?new)
    :prec (and (at briefcase ?loc) (location ?new) (not (= ?loc ?new))
          (not (in paycheck briefcase)) (in keys briefcase))
    :effect (and (at briefcase ?new) (not (at briefcase ?loc))
          (at keys ?new) (not (at keys ?loc)))

movebriefcaseboth (?loc ?new)
    :prec (and (at briefcase ?loc) (location ?new) (not (= ?loc ?new))
          (in paycheck briefcase) (in keys briefcase))
    :effect (and (at briefcase ?new) (not (at briefcase ?loc))
          (at paycheck ?new) (not (at paycheck ?loc))
          (at keys ?new) (not (at keys ?loc)))
```

Erweiterungen von Graphplan

- Der „Factored Expansion" Ansatz

```
movebriefcase (?loc ?new)
    :effect (when (and (at briefcase ?loc) (location ?new) (not (= ?loc ?new)))
                  (and (at briefcase ?new) (not (at briefcase ?loc))))
            (when (and (at briefcase ?loc) (location ?new) (not (= ?loc ?new))
                  (in paycheck briefcase))
                  (and (at paycheck ?new) (not (at paycheck ?loc))))
            (when (and (at briefcase ?loc) (location ?new) (not (= ?loc ?new))
                  (in keys briefcase))
                  (and (at keys ?new) (not (at keys ?loc))))
```

Erweiterungen von Graphplan

- Um mit konditionalen Effekten umzugehen gibt es verschiedene Möglichkeiten
- Der „Full Expansion" Ansatz ist nur für eine geringe Anzahl von Konditionen möglich, da er zu einem exponentiellen Zuwachs an STRIPS-Operatoren führt
- „Factored Expansion":
  - Vorteil: Leistungszuwachs
  - Nachteil: höhere Komplexität
    - es wird mit individuellen Aktionseffekten gearbeitet (nicht mehr mit den Aktionen selbst) → neue Ausschliessungseffekte: „induced effects"
    - Backward chaining Suche wird komplizierter: Algorithmus muss das Mittel der Konfrontation beherrschen
- Wir beschäftigen uns nun mit „Factored Expansion"

Erweiterungen von Graphplan

- Primitive Elemente: „components" / Komponenten
- Komponente ist Paar von
  - Antecedent (Disjunktion&Konjunktion von Literalen)
  - Consequent (Konjunktion von Literalen)
- Eine Aktion hat pro Effekt eine Komponente
- Vorbedingung der Komponente (Antecedent)
  - Konjunktion der Vorbedingungen von nicht konditionalem Effekt und jeweiligen konditionalem Effekt
- Gewöhnlich STRIPS-Aktionen besitzen nur eine Komponente

Erweiterungen von Graphplan

- Ein weiteres Beispiel

Intuitiv	full exp.	factored exp.
Operator1 pre: p effect: e_0 (when q e1) (when r e2)	Operator2a pre: p&~q&~r effect: e_0 Operator2b pre: p&~q&r effect: e_0&e_2 Operator2c pre: p&q&~r effect: e_0&e_1 Operator2d pre: p&q&r effect: e_0&e_1&e_2	Operator3 effect: (when p e_0) (when p&q e_1) (when p&r e_2)

Erweiterung von Graphplan

- Mutex-Relationen bei „Factored Expansion"
  - In erster Näherung bleibt alles beim alten, ausser dass wir Komponenten anstelle von Aktionen zum Graph hinzufügen. Deshalb gibt es genaugenommen keine Aktionslevels mehr sondern Komponentenlevels!
  - Für Propositionen (unverändert zum alten Graphplan):
    - Zwei Propositionen p und q schliessen sich aus im Level P_i, wenn alle Wege p zu erreichen und alle Wege q zu erreichen sich gegenseitig ausschliessen
  - Für Komponenten: Zwei Komponenten C_m und C_n aus Komponenten-Level A_i schliessen sich aus,
    - Interferenz/Inkonsistenz: .... falls sie von verschiedenen Aktionen stammen *und* der Consequent von C_n einen Consequent oder einen Antecedent von C_m löscht (oder umgekehrt)

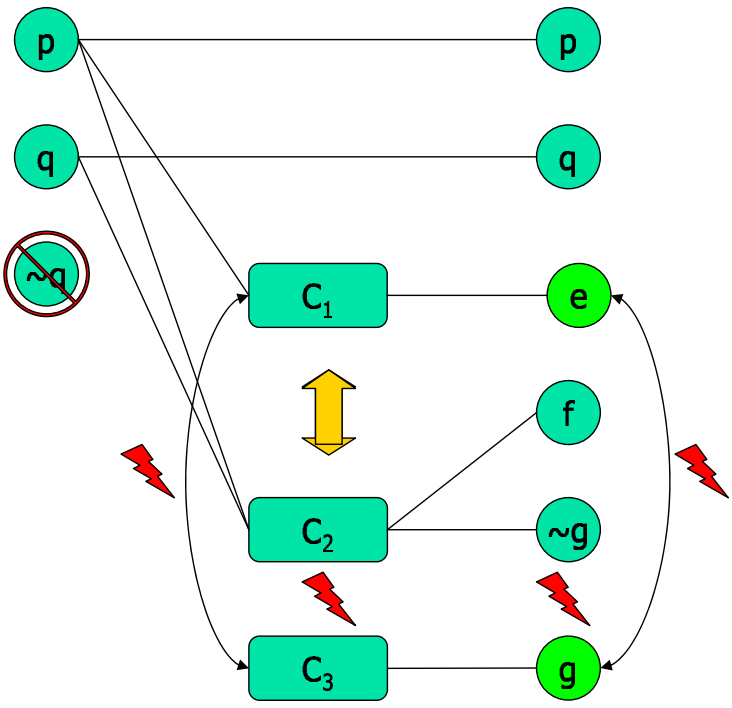
Erweiterungen von Graphplan

- Inkompatible Vorbedingungen: ... falls C_m und C_n Antecedents haben, die sich ausschließen auf Level P_i
- **Induzierte Komponente:** ... Falls C_k durch C_n induziert wird und C_k und C_m sich einander ausschließen.

- Dabei sagt man C_n induziert C_k, falls das Ausführen von C_n das gleichzeitige Ausführen von C_k bedingt, d.h. falls
  - C_n und C_k zur selben Aktion (gleiche Variablen Bindungen) gehören
  - C_n und C_k sich nicht einander ausschliessen
  - Die Negation des Antecedents von C_k auf Level P_i nicht erfüllt werden kann, sobald der Antecedent von C_n erfüllt ist.

Erweiterungen von Graphplan

I



Erweiterungen von Graphplan

- Wie wir am Beispiel sahen, muss unser Algroithmus die Möglichkeit der Konfrontation beherrschen.
  - C_m und C_n seien notwendige Komponenten für das Erreichen der Zielbedingungen
  - Per Induktion (C_n induziere C_k) wird die Mutex-Relation zwischen C_m und C_k auf C_n und C_m übertragen und somit das Erreichen des Zieles verhindert
- Konfrontation bedeutet, die Induktionsbeziehung aufzulösen, indem Teile der Vorbedingung von C_k verhindert werden, ohne dass Teile der Vorbedingungen von C_n und C_m verhindert werden.
- Der erweiterte Backchaining Algorithmus lautet nun wie folgt:

Erweiterung von Graphplan

- Backchaining Algorithm [Cond]:

1. Let SC_i={}, SC_i is the set of components selected at level i to support the goals at level i + 1.
2. For each goal g at level i + 1, choose (backtrack point) a supporting component C_s at level i that has g as an effect and that is nonmutex with the components in SC_i .
   i. If C_s is not already in SC_i , then add C_s to SC_i and add the preconditions of C_s to the goals for level i -1
3. For each pair of component C_s, C_t in SC_i, consider all pairs of components C_m, C_n where C_s possibly induces C_m and C_t possibly induces C_n. If C_m and C_n are mutex, then we must choose (backtrack point) one of C_m or C_n to confront
4. If i = 1, then the completed plan is the set of actions from which the components in SC_1 , SC_3 , SC_5 , . . . belong. Otherwise, reduce i by 2 and repeat.

Zusammenfassung

- Planen ist für das Handeln eines Agenten in einer komplexen Umgebung unumgänglich
- Die STRIPS-Repräsentation ermöglicht eine intuitive und gleichzeitig strenge Darstellung der Domäne
- Graphplan ist ein relativ neuer und vielversprechender Ansatz im Bereich des Planens
- Es ist möglich konditionale Effekte mit Graphplan zu behandelt. Eine effiziente Implementierung benutzt das Komponentenkonzept, beachtet explizit induzierte Ausschliessungen und verwendet das Mittel der Konfrontation.
- Ausblick: damit ist die Erweiterung von Graphplan noch längst nicht ausgeschöpft. Siehe z.B. nächste Woche im Seminar

Quellen / Literatur

- [Brewka] Materialien zur Vorlesung WBS 01/02
- [Cond] Weld, Daniel S. et al 1998. Conditional Effects in Graphplan
- Blum, A and Furst, M. 1995. Fast planning through planning graph analysis
- Weld, Daniel S. 1994. An introduction to least-commitment planning
- Weld, Daniel S. et al 1998. Extending graphplan to handle uncertainty & sensing actions