Reasonable & Forced
Goal Orderings
und ihr Gebrauch bei
Agenda-unterstütztes
Planen

Gegeben:

Menge von Zielen

Gesucht:

Ordnungsrelation über Teilmengen dieser Zielmenge

Ziel:

Effektives Planen bei voneinander abhängigen Teilzielen mit Hilfe einer Goal-Agenda

0 Gliederung

1. Ordnungs-Relationen zwischen atomaren Zielen
2. Methoden zur Berechnung von Goal Orderings
3. Benutzung einer Goal-Agenda
4. Empirische Ergebnisse

1 Ordnungs-Relationen zwischen atomaren Zielen

- Annahmen:
  - (O, I, G), A,B ϵ G atomare Ziele
  - Gültiger Plan (Sequenz von Aktionen) existiert
  - Planer verwirklicht A → s(A,¬B)

- 3 Möglichkeiten in s(A,¬B) :
  - Es existiert ein in s(A,¬B) ausführbarer Plan, der B verwirklicht, **ohne A zu löschen**.
  - Es exisitiert kein in s(A,¬B) ausführbarer Plan, der B verwirklicht. ->**Deadlock**
  - Es exisitiert nur ein Plan, der **A (vorübergehend) löscht**, bevor er B verwirklichen kann.

1.1 Forced Ordering

- Es exisitiert kein in s(A,¬B) ausführbarer Plan, der B verwirklicht. ->**Deadlock**

►*Planer ist gezwungen ("**forced**"), B vor oder gleichzeitig mit A zu verwirklichen.*

B ≤f A gdw. Für alle Zustände s(A,¬B) gilt: Es gibt keinen in s(A,¬B) ausführbaren Plan (Sequenz von Aktionen aus O), der B verwirklicht:

$$\forall s(A,\neg B): \neg\exists P^{O} : B \in result(s(A,\neg B), P^{O}) \quad (1)$$

1.2 Reasonable Ordering

- Es exisitiert nur ein Plan, der **A (vorübergehend) löschen** muss, bevor er B verwirklichen kann.

►*Es ist angemessener ("**reasonable**"), B vor oder gleichzeitig mit A zu verwirklichen, um nicht-optimale Pläne zu vermeiden.*

B ≤r A gdw. Für alle Zustände s(A,¬B) gilt: Es gibt keinen in s(A,¬B) ausführbaren Plan (Sequenz von Aktionen aus O_{A^-} alle Aktionen, die A nicht löschen) der B verwirklicht:

$$\forall s(A,\neg B): \neg\exists P^{O_{A^-}} : B \in result(s(A,\neg B), P^{O_{A^-}}) \quad (5)$$

1.3 Forced und Reasonable Orderings als Entscheidungsprobleme

- Angenommen es exisitieren Zustände s(A,¬B)

Gilt B ≤f A? ϵ PSPACE - hard
Gilt B ≤r A? ϵ PSPACE - hard

- Das Finden von Reasonable und Forced Ordering-Relationen zwischen atomaren Zielen ist mindestens so schwer wie das zu Grunde liegende Planungsproblem selbst!

???

2 Methoden zur Berechnung von Goal Orderings

- Definieren neuer Ordnungsrelationen, die in **Polynomialzeit** entscheidbar sind und möglichst eine **hinreichende Bedingung** für die Existenz von Reasonable oder Forced Ordering – Relationen darstellen.

- Approximation von Reasonable Goal Ordering:
  - ≤e „efficient ordering“, basierend auf **GRAPHPLAN**
  - ≤h „heuristic ordering“, basierend auf einer **heuristischen Methode**

2.1 Reasonable Goal Orderings basierend auf GRAPHPLAN

- Gegeben: $(O, I, \mathcal{G})$, A,B ϵ $\mathcal{G}$
- Ziel: mehr Wissen über s(A,¬B) erlangen
- Teilmenge F von Atomen bestimmen, für die gilt:

$$F \cap s(A, \neg B) = \varnothing$$

- GRAPHPLAN (GP) erstellen

False set von A bestimmen: alle Atome, die mit Aktionen aus O von I aus erreichbar sind und **nicht gleichzeitig mit A erfüllt** sein können, sich also mit A ausschließen:

$$F_{GP}^{A} := \{p \mid p \, mutex \, mit \, A \, wenn \, Plangraf \, "leveled \; off"\} \quad (7a)$$

2.1 Reasonable Goal Orderings basierend auf GRAPHPLAN

- <u>„Efficient Ordering“ ≤e</u>

B ≤e A gdw. Alle Aktionen des bzgl. A reduzierten Action set O_A (alle Aktionen, die A nicht löschen) fügen entweder B nicht hinzu oder, wenn doch, dann haben sie eine Vorbedingung, die im False set F_{GP}^{A} von A liegt:

$$\forall o_A \in O_A : B \in add(o_A) \Rightarrow pre(o_A) \cap F_{GP}^{A} \neq \varnothing \quad (6)$$

- B ≤e A => B ≤r A
- ≤e ist in Polynomialzeit entscheidbar

2.1.1 Beispiel ≤e: blocksworld

- I ={on(a,table), clear(a), on(b,table), clear(b), on(c,table), clear(c)}
- G ={on(a,b), on(b,c)}
- O ={

  push(x,y):

  holding(x) clear(y) → *ADD on(x,y) clear(x)*
  DEL holding(x) clear(y)

  pop(x,y):

  clear(x) on(x,y) → *ADD clear(y) holding(x)*
  DEL on(x,y) clear(x)

  }

2.2 Reasonable Goal Orderings basierend auf einer heuristischen Methode

- Direkte Analyse (DA) der Aktionen
- Plangraf nicht nötig!
- Anfangswert von F ($F \cap s(A, \neg B) = \varnothing$) sind alle Atome, die immer dann, wenn A hinzugefügt wird, gelöscht werden:

$$F_{DA}^{A} = \bigcap_{o \in O,\, A \in add\,(o)} del\,(o) \qquad (7b)$$

- approximierte **Fixpunkt-Reduktion** von F und von O_A: Alle Atome, die mit Aktionen aus O_A dann doch noch erreichbar sind - im Sinne von **possibly achievable** (8) –, werden aus F entfernt.

2.2.1 Fixpunkt-Reduktion

$$
\begin{aligned}
&F^* := F^A_{DA} \\
&O^* := O_A - \left\{o \mid F^* \cap pre(o) \neq \emptyset\right\} \\
&fixp_erreicht = false \\
&while(\neg fixp_erreicht)\{ \\
&\qquad fixp_erreicht = true \\
&\qquad for\ f \in F^*\{ \\
&\qquad\qquad if\ pA(f, O^*)\ then\{ \\
&\qquad\qquad\qquad F^* := F^* - \{f\} \\
&\qquad\qquad\qquad O^* := O_A - \{o \mid F^* \cap pre(o) \neq \emptyset\} \\
&\qquad\qquad\qquad fixp_erreicht := false \\
&\qquad\qquad \} \\
&\qquad \} \\
&\} \ return\ F^*, O^*
\end{aligned}
$$

2.2.1 Fixpunkt-Reduktion

- Ziel:
  - Am Ende befinden sich in F* nur noch die **Atome, die nicht von s(A,¬B) aus erreichbar sind ohne A zu löschen**.
  - Im reduzierten Action set O* befinden sich am Ende alle die **Aktionen, die nach s(A,¬B) anwendbar sind, ohne A zu löschen**.
  - Nun kann sicher gesagt werden, ob ein Ziel mit diesen Akitonen noch erreicht werden kann. Wenn nicht:

-> reasonable ordering!!

2.2.2 Heuristic Ordering

- „heuristic ordering“

B ≤h A gdw.B ist nicht erreichbar – im Sinne von possibly achievable – mit Hilfe von Aktionen des bzgl. A reduzierten Action set O* (Resultat der Fixpunkt-Berechnung):

$$\neg pA(B, O^*) \qquad (9)$$

- B ≤h A => B ≤r A

2.2.3 Beispiel Fixpunkt-Reduktion

- op1: → add{A} del{C,D}
- op2: → add{A,C} del{D}
- op3: {C} → add{D}
- op4: {D} → add{B}

- Gilt B ≤h A?

2.2.4 Beispiel ≤h - blocksworld

- on(a,b) ≤h on(b,c) ?
- Nach Fixpunkt-Reduktion bzgl. on(b,c):
  F* = {holding(b), clear(c)}
  O* = {push(a,b), push(c,a), push(c,b), pop(a,b), pop(a,c), pop(b,a)}
- Test: pA(on(a,b),O*)?
  → on(a,b) ϵ add(push(a,b))
  → pre(push(a,b)) =
  holding(a) ϵ add(pop(a,b))
  clear(b) ϵ add(pop(a,b))
- pA(on(a,b),O*) → ¬ (on(a,b) ≤h on(b,c))

2.2.4 Beispiel ≤h - blocksworld

- on(b,c) ≤h on(a,b) ?
- Nach Fixpunkt-Reduktion bzgl. on(a,b):
  F* = {holding(a), clear(b)}
  O* = {push(b,c), push(b,a), push(c,a), pop(a,c), pop(c,a), pop(c,b)}
- Test: pA(on(b,c),O*)?
  → on(b,c) ϵ add(push(b,c))
  → pre(push(b,c)) =
  clear(c) ϵ add(pop(a,c))
  holding(b) ϵ add(pop(b,?))
- ¬ pA(on(b,c),O*) → on(b,c) ≤h on(a,b)

2.3 Forced Goal Orderings

- Keine Bemühungen, Approximationen speziell für Forced Goal Orderings zu finden
- Gründe:
  - ≤r Abschätzung für ≤f
    => ≤e und ≤h grobe Abschätzung für ≤f
  - Viele Benchmark-Planprobleme beinhalten ausschließlich invertierbare Aktionen. D. h., jede Aktion lässt sich rückgängig machen. Es gibt keine „Deadlocks“ (Zustände, von denen aus keine Aktionen mehr anwendbar sind, die zum Ziel führen).
    -> „invertible Probleme“
  - Nicht-triviales forced-ordering B ≤f A impliziert jedoch die Existenz von Deadlocks.

=> Die meisten interessierenden Probleme sind deadlock- und damit forced ordering-frei

3 Benutzung einer Goal Agenda

- Zerlegung der Zielmenge in disjunkte Teilmengen
- Planer läuft auf diesen sequentiell geordneten Teilproblemen
- Gesamtplan ist dann Verkettung der Teilpläne

3.1 Erstellen einer Goal Agenda

1. Zielanalyse
   - Jedes Paar A,B ϵ $\mathcal{G}$ wird auf Ordnungsrelation hin untersucht:

     A ≤ B oder B ≤ A oder beides oder keines
2. Zielmenge $\mathcal{G}$ in kleinere Mengen entsprechend diesen Relationen zerlegen und entsprechend den Relationen ordnen

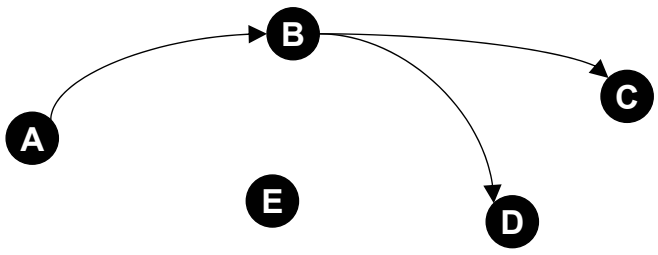
Ziel: Sequenz von Zielmengen (Goal-Agenda G)

G = $G_1,\ldots,G_n \; mit \; \bigcup_{i=1}^{n} G_i = \mathcal{G}, \; G_i \cap G_j = \varnothing \quad i \neq j, \; 1 \leq i, j \leq n$ (10)

3.1 Goal Graph G

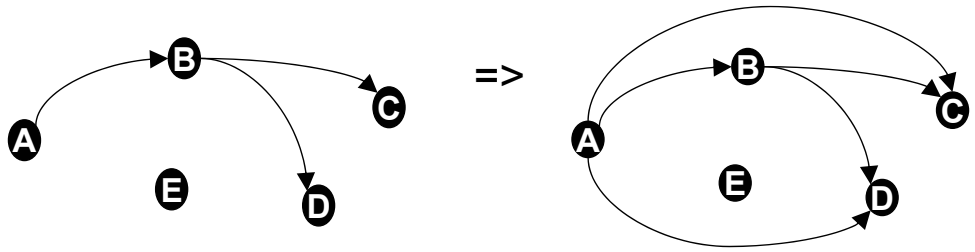
- Gerichteter Graph G:=(V,E)
- V:= $\mathcal{G}$
- E:={(A,B) ϵ $\mathcal{G}$ x $\mathcal{G}$ | A ≤ B}

- Bsp: $\mathcal{G}$ = {A,B,C,D,E}, A ≤ B, B ≤ C, B ≤ D



3.2 Berechnung der Goal Agenda

1. Transitive Hülle G* von G berechnen



2. Für jeden Knoten A die in G* ausgehenden und eingehenden Kanten zählen
   - G-sep: Menge der unverbundenen Knoten
   - Übrige Mengen beinhalten alle Knoten mit dem selben Grad d(A) = A_in – A_out
3. Ordnen dieser Mengen nach ansteigenden Grad => gewünschte Sequenz:

-3		-1		2	G-sep
{A}	→	{B}	→	{C,D}	{E}

3.2 Erweiterung von Goal Orderings für Zielmengen

- Geg.: $(O, I, \mathcal{G})$, $\{A_1,...,A_n\} \subseteq \mathcal{G}$, $\{B_1,...,B_k\} \subseteq \mathcal{G}$
- Ziel: Erweiterung von ≤e auf ≤E

$$F_{GP}^{\{A_1,...,A_n\}} \qquad O_{\{A_1,...A_n\}} \qquad \text{(11a,b)}$$

und ≤h auf ≤H

$$F_{DA}^{\{A_1,...,A_n\}} \qquad O^{\star} \qquad \text{(11c,d)}$$

3.2.1 Ordering ≤E

$\{B_1,...,B_k\} \leq E \{A_1,...,A_n\}$ gdw. ein B_j von keiner Aktion aus dem reduzierten Action-set bzgl. der Menge A hinzugefügt werden kann oder, wenn doch, dann liegt eine Vorbedingung der Aktion, die B hinzufügt, im False set der Menge A:

$$\exists j \in \{1,...,k\} : \forall o \in O_{\{A_1,...,A_n\}} : B_j \in add(o) \Rightarrow pre(o) \cap F_{GP}^{\{A_1,...,A_n\}} \neq \varnothing$$

3.2.1 Ordering ≤H

$\{B_1,...,B_k\} \leq H \{A_1,...,A_n\}$ gdw. Es existiert ein B_j, das mit Aktionen aus $O^\star$ nicht erreichbar (pA) ist:

$$\exists j \in \{1,...,k\} : \neg pA(B_j, O^*)$$

3.3 Der Agenda-driven planning Algorithmus

- Geg.: (O,I,G), Goal Agenda $G_1,\ldots,G_n$, $G_i \subseteq G$
- Idee: Planer wird initialisiert mit $I_1 = I$ $G_1 = G_1$
- Planer löst Problem (O, I_1, G_1)
- Führt Plan P_1 aus
- Neuer Initialzustand $I_2 = result(I_1, P_1)$
- Neues Problem (O, I_2, G_2) $G_2 = \bigcup_{j=1}^{i} G_1 \cup G_2$
- Allgemein: $G_i = \bigcup_{j=1}^{i} G_j$, $G = G_k = \bigcup_{j=1}^{k} G_j$
- Plan für (O,I,G) = P_1 , P_2 , … , P_k

3.3 Der Agenda-driven planning Algorithmus

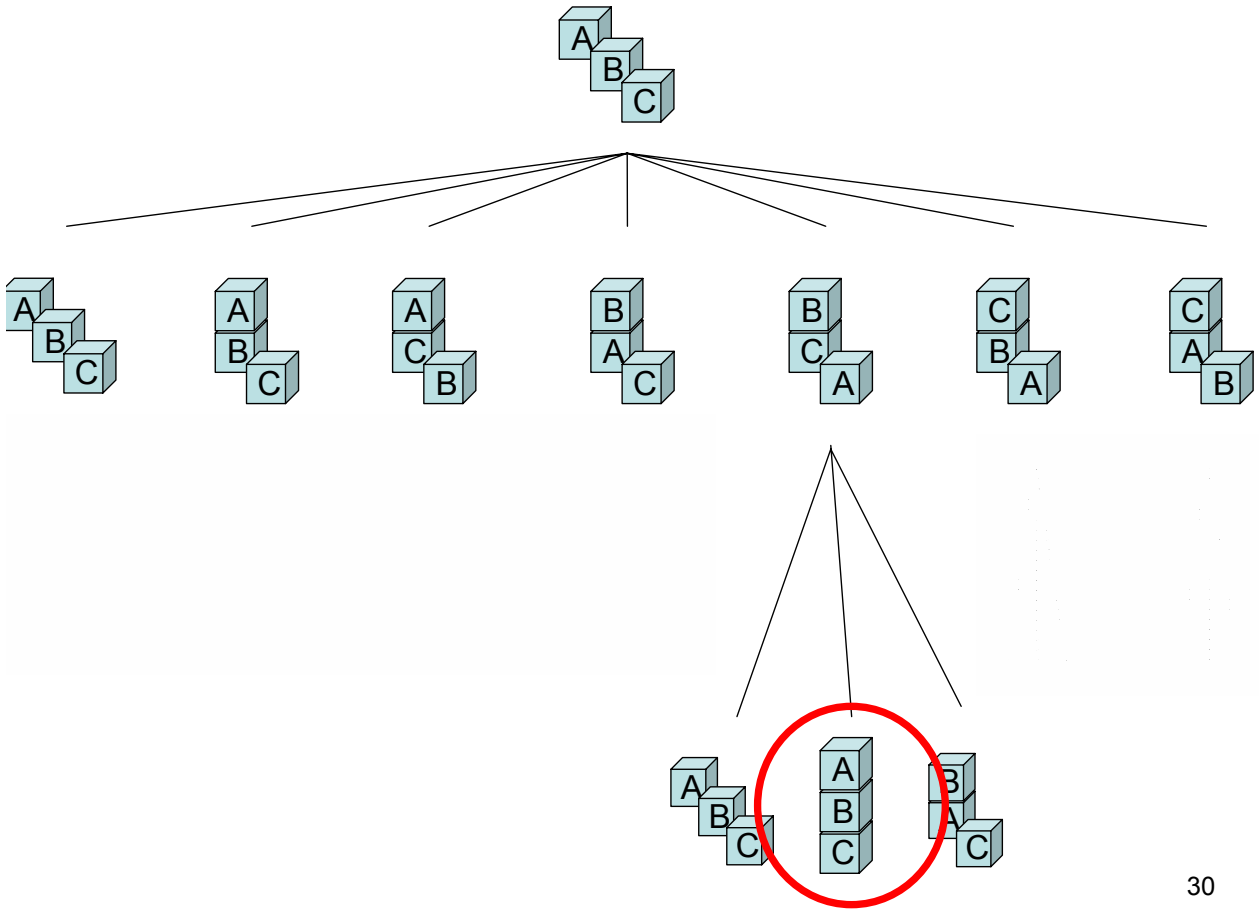
- No-ops-first (GRAPHPLAN-feature)
  (Algorithmus versucht zuerst Ziele zu erreichen, indem er sie, wenn möglich, auf „wahr“ lässt, sie also wenn sie bereits erreicht sind nicht zu zerstören)
- Reasonable ordering bietet Algorithmus bereits eine solche Sortierung, so dass dieses feature optimal ausgenutzt werden kann
- Was den Algorithmus also auszeichnet, ist das Berechnen einer Sequenz sondierter Teilpläne, die einen Zustandsübergang von I nach G liefern („Soundness“: „Korrektheit“)
- „Vollständigkeit“ bei Problemen ohne Deadlocks gewährleistet.

3.3 Der Agenda-driven planning Algorithmus

- Vollständigkeit dennoch nicht allgemein garantierbar:

O= { op1:{C} → add {B} del {D}
op2:{D} → add {E}
op3:{E} → add {F}
op4:{F} → add {A} } $\mathcal{I}$ = {C,D}, $\mathcal{G}$ = {A,B}

A ≤h B ?	B ≤h A ?
F* = {D}	F = Ø
O* = {op3, op4}	O* = Ø
pA(A,O*)? Ja, op4	pA(B,O*)? Nein!
⇒A ≤h B nicht ableitbar!	⇒B ≤h A ableitbar!
Falsch! A ≤f B!	Falsch! B nach A erreichbar!



4 Empirische Ergebnisse

- IPP –Planer
- Beide Methoden implementiert als sogenannten Goal Agenda Manager (GAM)
- IPP-Planning-Algorithmus: Lösen von Teilproblemen und Konkatenation von Teilplänen ergibt Gesamtplan
- IPP-domain-Kollektion: 48 Domains
- Goal orderings ließen sich für 10 Domains ableiten
- GAM fand fast alle reasonable orderings

4.1 Vergleich ≤e und ≤h in Blocksworld-Domain

Problem	#actions	#agenda entries	CPU(≤e)	CPU(≤h)
Bw_large_a	162	1	0.69	0.07
Bw_large_b	242	5	1.45	0.11
Bw_large_c	450	7	4.85	0.22
Bw_large_d	722	11	14.18	0.35
Bw_large_e	722	11	12.95	0.35
Bw_large_f	1250	6	44.93	0.58
Bw_large_g	1800	9	97.11	0.88
Stack_20	800	19	6.91	0.36
Stack_40	3200	39	160.00	1.74
Stack_60	7200	59	840.42	4.85
Stack_80	12800	79	-	11.38

4.1 Vergleich ≤e und ≤h in anderen Domain

Problem (Domain)	#actions	#agenda entries	CPU(≤e)	CPU(≤h)
Bull (Bulldozer)	61	2/1	0.09	0.03
Glass1 (glassworld)	26	2/1	0.02	0.01
Glass2 (glassworld)	114	2/1	0.19	0.09
Glass3 (glassworld)	122	2/1	0.22	0.09
Sched6 (scheduling)	104	1/4	1.0	0.12
Wood1 (woodshop)	15	1/3	0.03	0.01
Wood2 (woodshop)	15	6/5	0.03	0.01
Wood3 (woodshop)	43	6/5	0.14	0.06

4.2 Einfluss von Goal Orderings auf die Performance von IPP in Verbindung mit RIFO –Blocks-World

- RIFO: heuristische Analyse-Methode, die IPP befähigt, irrelevante Aktionen und irrelevante Initial-Fakten aus dem Planning-Problem zu entfernen

SATPLAN	#actions	Plan length	IPP	+GAM	+GAM+RIFO
Bw_large_a	162	12	0.7	0.74	0.58
Bw_large_b	242	22(18)	26.71	0.86	0.55
Bw_large_c	450	48	-	7.34	2.42
Bw_large_d	722	54	-	11.62	3.74
Bw_large_e	722	52	-	11.14	3.99
Bw_large_f	1250	90	-	-	-
Bw_large_g	1800	84	-	-	117.56

4.2 Einfluss von Goal Orderings auf die Performance von IPP in Verbindung mit RIFO – Türme von Hanoi

Scheiben	#actions	IPP	+GAM
2	21	0.02	0.02
3	48	0.08	0.07
4	90	0.33	0.25
5	150	1.57	3.10
6	231	9.71	88.45
7	336	69.44	2339.94

- Türme von Hanoi EINZIGE gefundene Domäne, wo Erweiterung von IPP um GAM zu schlechteren Ergebnissen führte!!

5 Zusammenfassung

- Ziel war: Planer mit zusätzlichen Informationen über die Reihenfolge der Ziele zu unterstützen
- Berechnung in Polynomialzeit
- Entwickelte Methoden für STRIPS und ADL
- „Soundness“ immer und „Completeness“ bei Deadlock-freien Problemen (also für reasonable ordering) gewährleistet
- ≤e m. H. Plangraf, ≤h m. H. Direkte Analyse der Aktionen
- ≤e schlechtere Performance als ≤h ansonsten gleich gut
- Goal Agenda Manager verbessert i. A. signifikant die Leistung von IPP

Literatur

- Koehler, Jana; Hoffmann, Jörg: On Reasonable and Forced Goal Orderings and their Use in an Agenda-Driven Planning Algorithm

Forced Goal Ordering: $B \leq_f A \Leftrightarrow \forall s(A,\neg B): \neg\exists P^O : B \in result(s(A,\neg B), P^O)$ (1)

Plan: $P^O = \langle o_1,...,o_n \rangle, o \in O$ (2)

result: $result(s,o) = \begin{cases} (s \cup add(o)) - del(o) & if\ pre(o) \subseteq s \\ s & otherwise \end{cases}$

$result(s, \langle o_1,...,o_n \rangle) := result(result(s, \langle o\ ,...o_{n-1} \rangle), o_n)$ (3)

reduziertes Action set bzgl. A: $O_A = \{o \in O \mid A \notin del(o)\}$ (4)

Reasonable Goal Ordering: $B \leq_r A \Leftrightarrow \forall s(A,\neg B): \neg\exists P^{O_A} : B \in result\,(s(A,\neg B), P^{O_A})$ (5)

Efficient Goal Ordering: $B \leq_e A \Leftrightarrow \forall\, o_A \in O_A : B \in add\,(o_A) \Rightarrow pre(o_A) \cap F_{GP}^A \neq \emptyset$ (6)

False set von A (m. H. von GRAFPL.): $F_{GP}^A := \{p \mid p\ mutex\ mit\ A\ wenn\ Plangraf\ "leveled\ off"\}$ (7a)

False set von A (m. H. von Direkte Analyse): $F_{DA}^A := \bigcap_{o \in O, A \in add(o)} del(o)$ (7b)

Possibly achievable: $pA(p,O) \Leftrightarrow \exists o \in O : p \in add(o) \wedge \forall p' \in pre(o) : \exists o' \in O : p' \in add(o')$ (8)
(Alle Atome, die add-Effekte von Aktionen sind, deren Vorbedingungen ebenfalls add-Effekte sind, heißen possibly achievable)

Fixpunkt-Reduktion:

```
F* := F_DA^A
O* := O_A − {o | F* ∩ pre(o) ≠ ∅}
fixp_erreicht = false
while(¬fixp_erreicht){
        fixp_erreicht = true
        for f ∈ F* {
                if pA(f, O*) then{
                        F* := F* − {f}
                        O* := O_A − {o | F* ∩ pre(o) ≠ ∅}
                        fixp_erreicht := false
                }
        }
}return F*, O*
```

Heuristic Goal Ordering: $B \leq_h A \Leftrightarrow \neg pA(B, O^*)$ (9)

Goal Agenda: $G_1,...,G_n\ mit\ \bigcup_{i=1}^{n} G_i = G,\ G_i \cap G_j = \emptyset \qquad i \neq j,\ 1 \leq i,j \leq n$ (10)

False set (≤E): $F_{GP}^{\{A_1,...,A_n\}} := \{p \mid p\ ist\ mutex\ mit\ wenigstens\ einem\ A_i\ wenn\ Plangraph\ "leveled\ off"\}$ (11a)

Reduziertes Action set (≤E): $O_{\{A_1,...A_n\}} = \{o \in O \mid \forall i \in \{1,...n\} : A_i \notin del(o)\}$ (11b)

False set (≤H): $F_{DA}^{\{A_1,...,A_n\}} := \bigcup_i F_{DA}^{A_i}$ (11c)

$O^* := O_{\{A_1,...,A_n\}} - \{o \mid \exists i \in \{1,...,n\} : A_i \in del(o) \vee F_{DA}^{\{A_1,...,A_n\}} \cap pre(o) \neq \emptyset\}$ (11d)

Efficient Goal set Ordering:

$\{B_1,...,B_k\} \leq_E \{A_1,...,A_n\} \Leftrightarrow \exists j \in \{1,...,k\} : \forall o \in O_{\{A_1,...,A_n\}} : B_j \in add(o) \Rightarrow pre(o) \cap F_{GP}^{\{A_1,...,A_n\}} \neq \emptyset$ (12)

Beispielrechnung zu 2.1.1 Beispiel ≤e: blocksworld

Test 1: Gilt on(a,b) ≤e on(b,c)?

1. False set von on(b,c):
   $$F_{GP}^{on(b,c)} = \{clear(c), hold(b), hold(c), on(b,table), on(b,a), on(a,c), on(c,b)\}$$
   Wenn b auf c steht, kann nicht gleichzeitig c frei sein, b oder c gehalten werden, b auf dem Tisch stehen, b auf a, a auf c oder c auf b stehen.

2. Reduziertes action set von on(b,c):
   $$O_{on(b,c)} = O - \{pop(b,c)\}$$
   Alle anwendbaren Aktionen aus O, außer b von c runter nehmen, zerstören den Zustand, b steht auf c, nicht.

3. Testen, ob on(a,b) ≤e on(b,c) gilt:
   - Welche Aktionen aus $O_{on(b,c)}$ haben on(a,b) in ihrer add-Liste?
     - push(a,b) Wenn b auf c steht, lässt sich on(a,b) dadurch verwirklichen, dass a auf b gestellt wird.
   - Hat push(a,b) Vorbedingungen, die nicht gleichzeitig mit on(b,c) gelten können, also im False Set von on(b,c) sind?
     - Pre(push(a,b)) = {hold(a), clear(b)} $\not\subset$ $F_{GP}^{on(b,c)}$, also NEIN!
       schließen sich — Vorbedingungen der Aktion(en), die on(a,b) erfüllt, nicht mit on(b,c) aus.
   - on(a,b) ≤e on(b,c) NICHT ableitbar!

Test 2: Gilt on(b,c) ≤e on(a,b)?

1. False set von on(a,b):
   $$F_{GP}^{on(a,b)} = \{clear(a), hold(b), hold(a), on(a,table), on(b,a), on(a,c), on(c,b)\}$$

2. Reduziertes action set von on(b,c):
   $$O_{on(a,b)} = O - \{pop(a,b)\}$$

3. Testen, ob on(b,c) ≤e on(a,b) gilt:
   - Welche Aktionen aus $O_{on(a,b)}$ haben on(b,c) in ihrer add-Liste?
     - push(b,c)
   - Hat push(b,c) Vorbedingungen, die nicht gleichzeitig mit on(a,b) gelten können, also im False Set von on(a,b) sind?
     - Pre(push(b,c)) = {hold(b), clear(c)} → hold(b) ∈ $F_{GP}^{on(b,c)}$, also JA!
   - on(b,c) ≤e on(a,b) ableitbar!

Fazit: es ist effektiver, erst b auf c zu stellen und danach a auf b (on(b,c) ≤e on(a,b)), da, wenn zuerst on(a,b) gilt, dieses zwischenzeitlich wieder gelöscht werden müsste.

Beispielrechnung zu 2.2.3 Beispiel Fixpunktreduktion

Gilt B $\leq_h$ A?

1. FalseSet $F^* = F_{DA}^A = \bigcap_{o \in O, A \in abb(o)} del(o) = \{D\}$

2. Reduziertes ActionSet von A $O^* = \{op1, op2, op3\}$

3. $pA(D, O^*)$? Ist D mit Aktionen aus O^* possibly achievable? JA!

$\Rightarrow 1. F^* = F^* - \{D\} = \varnothing$

F * beinhaltet die Atome, die nicht von A aus erreichbar sind ohne A zu löschen

$\Rightarrow 2. O^* = \{op1, op2, op3, op4\}$

O * beinhaltet die von A aus anwendbaren Aktionen, die A nicht zerstören

$\Rightarrow B \not\leq_h A$, da B mit Aktionen aus O * erfüllbar ist