

Einführung in die Künstliche Intelligenz

SoSe 19

Prof. Dr. J. Fürnkranz, Prof. Dr. K. Kersting



Beispiellösung für das 4. Übungsblatt

Aufgabe 1 Planen, STRIPS

In dieser Beispiellösung wurde eine sehr direkte Modellierung benutzt.

a)

pos(p1)	% p1 ist eine Position	box(k)	% k ist eine Kiste
pos(p2)	% p2 ist eine Position	at(k,p3)	% Kiste k ist auf Position p3
pos(p3)	% p3 ist eine Position	toy(s1)	% s1 ist ein Spielzeug
monkey(a)	% a ist ein Affe	toy(s2)	% s2 ist ein Spielzeug
at(a,p1)	% Affe a ist auf Position p1	toy(s3)	% s3 ist ein Spielzeug
hungry(a)	% Affe a ist hungrig	at(s1,p1)	% Spielzeug s1 ist auf Position p1
on_floor(a)	% Affe a ist auf dem Boden	at(s2,p2)	% Spielzeug s2 ist auf Position p2
banana(b)	% b ist eine Banane	at(s3,p3)	% Spielzeug s3 ist auf Position p3
at(b,p2)	% Banane b ist auf Position p2		

b)

action: go(A,P)
preconditions: monkey(A), at(A,Q), on_floor(A), pos(Q), pos(P)
add: at(A,P)
delete: at(A,Q)

action: push(A,K,P)
preconditions: monkey(A), box(K), at(A,Q), at(K,Q), on_floor(A), pos(Q), pos(P)
add: at(A,P), at(K,P)
delete: at(A,Q), at(K,Q)

action: throw(A,S,P)
preconditions: monkey(A), toy(S), at(A,Q), at(S,Q), on_floor(S), pos(Q), pos(P)
add: at(S,P)
delete: at(S,Q)

action: up(A,K)
preconditions: monkey(A), box(K), at(A,P), at(K,P), on_floor(A), pos(P)
add: on_box(A)
delete: on_floor(A)

action: down(A)
preconditions: monkey(A), on_box(A)
add: on_floor(A)
delete: on_box(A)

action: eat(A,B)
preconditions: monkey(A), banana(B), at(A,P), at(B,P), on_box(A), pos(P)
add: full(A)
delete: hungry(A), at(B,P)

c) Die Formulierung des Ziels lautet: full(a)

Der kürzeste Plan hierfür ist:

go(a, p3)

push(a, k, p2)

up(a, k)

eat(a, b)

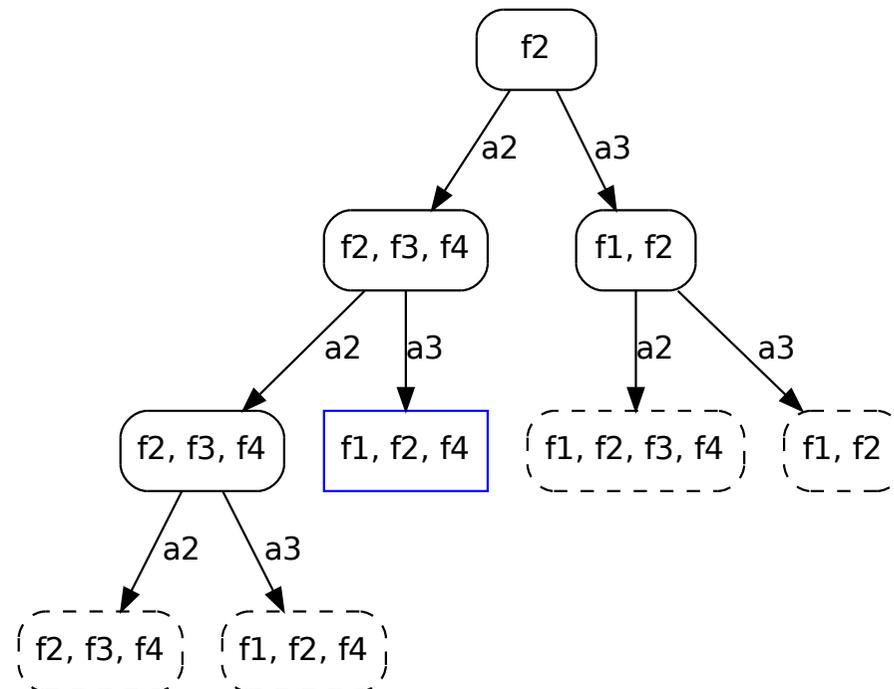
Die folgende Tabelle veranschaulicht die geltenden Fakten vor bzw. nach der Abarbeitung einer Aktion.

Start	go(a, p3)	push(a, k, p2)	up(a, k)	eat(a, b)
pos(p1)	pos(p1)	pos(p1)	pos(p1)	pos(p1)
pos(p2)	pos(p2)	pos(p2)	pos(p2)	pos(p2)
pos(p3)	pos(p3)	pos(p3)	pos(p3)	pos(p3)
monkey(a)	monkey(a)	monkey(a)	monkey(a)	monkey(a)
at(a, p1)	at(a, p3)	at(a, p2)	at(a, p2)	at(a, p2)
hungry(a)	hungry(a)	hungry(a)	hungry(a)	full(a)
on_floor(a)	on_floor(a)	on_floor(a)	on_box(a)	on_box(a)
banana(b)	banana(b)	banana(b)	banana(b)	banana(b)
at(b, p2)	at(b, p2)	at(b, p2)	at(b, p2)	-
box(k)	box(k)	box(k)	box(k)	box(k)
at(k, p3)	at(k, p3)	at(k, p2)	at(k, p2)	at(k, p2)
toy(s1)	toy(s1)	toy(s1)	toy(s1)	toy(s1)
toy(s2)	toy(s2)	toy(s2)	toy(s2)	toy(s2)
toy(s3)	toy(s3)	toy(s3)	toy(s3)	toy(s3)
at(s1, p1)	at(s1, p1)	at(s1, p1)	at(s1, p1)	at(s1, p1)
at(s2, p2)	at(s2, p2)	at(s2, p2)	at(s2, p2)	at(s2, p2)
at(s3, p3)	at(s3, p3)	at(s3, p3)	at(s3, p3)	at(s3, p3)

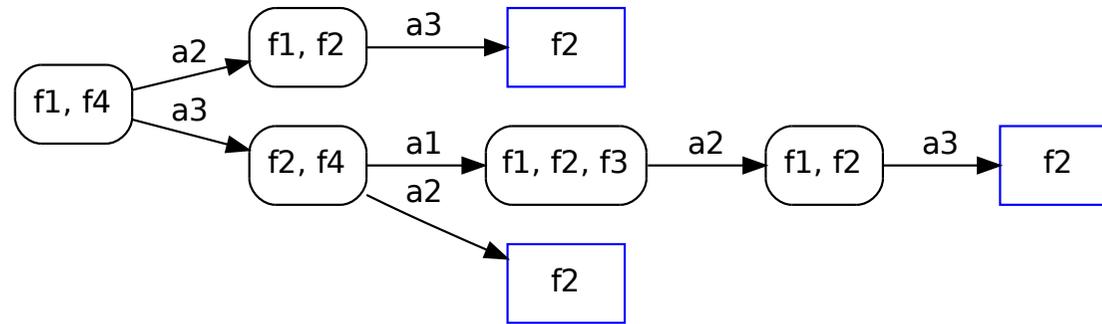
Aufgabe 2 Vorwärts-, Rückwärtsplanen

- a) Je nach Implementierung der Breitensuche werden die gestrichelt umrandeten Knoten erzeugt oder nicht, d.h. falls der *Goal-Test* unmittelbar nach Expandierung eines Knotens angewendet wird, werden die gestrichelt umrandeten Knoten nicht erzeugt.

Es wird der Plan (a_2, a_3) gefunden.



- b) Es werden folgende Pläne gefunden: (a_3, a_2) , (a_3, a_2, a_1, a_3) und (a_2, a_3)



Aufgabe 3 Partial-Order Planning

- a) Im Partial-Order Planning Algorithmus finden verschiedene Selektionen, z.B. die Wahl der nächsten open precondition oder die Wahl der nächsten Aktion für die Verfeinerung des Plans, statt, die auf den Vorlesungs-Folien und in der Aufgabenstellung nicht näher erläutert bzw. spezifiziert werden. Hier war es erlaubt, eine mehr oder weniger beliebige gültige Selektion vorzunehmen. Für diese Beispiellösung wurde eine Abarbeitungsfolge benutzt, die ein Konflikt beinhaltet und nicht zu aufwendig ist. In blau werden die Änderungen dargestellt.

Der Anfangszustand sieht folgendermaßen aus:

```
Actions = {Start, Finish}
Orderings = {Start < Finish}
Causal Links = {}
Open preconditions = {on(a,table), on(b,table), on(c,table), on(d,table)}
```

Refining - folgende mögliche Aktionen stehen zur Auswahl:

putdown(a), putdown(b), putdown(c), putdown(d), Start

Es wird **putdown(a)** ausgewählt.

```
Actions = {Start, Finish, putdown(a)}
Orderings = {Start < Finish,
             Start < putdown(a), putdown(a) < Finish}
Causal Links = {putdown(a) → on(a,table) → Finish}
Open preconditions = {on(b,table), on(c,table), on(d,table), holding(a)}
```

Refining - folgende mögliche Aktionen stehen zur Auswahl:

putdown(b), putdown(c), putdown(d), Start, unstack(a,B)

Es wird unstack(a,B) ausgewählt. Hier handelt es sich um einen Fall, in dem Variablen ungebunden vorkommen. Wir wählen den Ansatz, jetzt eine beliebige Substitution vorzunehmen, nämlich {b/B} (also **unstack(a,b)**).

```
Actions = {Start, Finish, putdown(a), unstack(a,b)}
Orderings = {Start < Finish,
             Start < putdown(a), putdown(a) < Finish,
             Start < unstack(a,b), unstack(a,b) < Finish,
             unstack(a,b) < putdown(a)}
Causal Links = {putdown(a) → on(a,table) → Finish,
                unstack(a,b) → holding(a) → putdown(a)}
Open preconditions = {on(b,table), on(c,table), on(d,table), handempty, block(a), block(b), on(a,b)}
```

Refining - folgende mögliche Aktionen stehen zur Auswahl:

putdown(b), putdown(c), putdown(d), Start

Es wird **Start** ausgewählt.

```
Actions = {Start, Finish, putdown(a), unstack(a,b) }
Orderings = {Start < Finish,
             Start < putdown(a), putdown(a) < Finish,
             Start < unstack(a,b), unstack(a,b) < Finish,
             unstack(a,b) < putdown(a)}
Causal Links = {putdown(a) → on(table(a) → Finish,
                unstack(a,b) → holding(a) → putdown(a),
                Start → handempty → unstack(a,b)}
Open preconditions = {on(c,table)}
```

Refining - folgende mögliche Aktionen stehen zur Auswahl:

putdown(c)

Die Wahl ist eindeutig und es wird **putdown(c)** ausgewählt.

```
Actions = {Start, Finish, putdown(a), unstack(a,b), putdown(c) }
Orderings = {Start < Finish,
             Start < putdown(a), putdown(a) < Finish,
             Start < unstack(a,b), unstack(a,b) < Finish,
             unstack(a,b) < putdown(a),
             Start < putdown(c), putdown(c) < Finish}
Causal Links = {putdown(a) → on(table(a)) → Finish,
                unstack(a,b) → holding(a) → putdown(a),
                Start → handempty → unstack(a,b),
                putdown(c) → on(c, table) → Finish}
Open preconditions = {holding(c)}
```

Refining - folgende mögliche Aktionen stehen zur Auswahl:

unstack(c,B)

Die Wahl ist eindeutig und es wird hier unstack(c,B) mit der Substitution $\{d/B\}$, also unstack(c,d) ausgewählt.

Es wird nun in zwei Schritten das Refining dargestellt, da hier ein Konflikt auftritt. Im folgenden ist der Zustand vor der Konflikt-Auflösung dargestellt.

```
Actions = {Start, Finish, putdown(a), unstack(a,b), putdown(c), unstack(c,d) }
```

```
Orderings = {Start < Finish,  
             Start < putdown(a), putdown(a) < Finish,  
             Start < unstack(a,b), unstack(a,b) < Finish,  
             unstack(a,b) < putdown(a),  
             Start < putdown(c), putdown(c) < Finish,  
             Start < unstack(c,d), unstack(c,d) < Finish}
```

```
Causal Links = {putdown(a) → on(table(a) → Finish,  
                unstack(a,b) → holding(a) → putdown(a),  
                Start → handempty → unstack(a,b),  
                putdown(c) → on(c, table) → Finish,  
                unstack(c,d) → holding(c) → putdown(c)}
```

```
Open preconditions = {handempty, block(c), block(d), on(c,d)}
```

Die Aktion `unstack(c,d)` verursacht ein Konflikt mit dem kausalen Link `Start → handempty → unstack(a,b)`, da es `¬handempty` als Effekt hat und zwischen `Start` und `unstack(a,b)` eintreten kann. Dieser Konflikt kann nur dadurch gelöst werden, in dem das Ordering `unstack(a,b) < unstack(c,d)` eingefügt wird. Es gilt nun also:

```
Actions = {Start, Finish, putdown(a), unstack(a, b), putdown(c), unstack(c, d) }
Orderings = {Start < Finish,
             Start < putdown(a), putdown(a) < Finish,
             Start < unstack(a, b), unstack(a, b) < Finish,
             unstack(a, b) < putdown(a),
             Start < putdown(c), putdown(c) < Finish,
             Start < unstack(c, d), unstack(c, d) < Finish,
             unstack(a, b) < unstack(c, d)}
Causal Links = {putdown(a) → on(table(a) → Finish,
                unstack(a, b) → holding(a) → putdown(a),
                Start → handempty → unstack(a, b),
                putdown(c) → on(c, table) → Finish,
                unstack(c, d) → holding(c) → putdown(c)}
Open preconditions = {handempty, block(c), block(d), on(c, d)}
```

Refining - folgende mögliche Aktionen stehen zur Auswahl (*):

Start, putdown(X)

Es wird Start gewählt.

```
Actions = {Start, Finish, putdown(a), unstack(a,b), putdown(c), unstack(c,d) }
```

```
Orderings = {Start < Finish,  
             Start < putdown(a), putdown(a) < Finish,  
             Start < unstack(a,b), unstack(a,b) < Finish,  
             unstack(a,b) < putdown(a),  
             Start < putdown(c), putdown(c) < Finish,  
             Start < unstack(c,d), unstack(c,d) < Finish,  
             unstack(a,b) < unstack(c,d)}
```

```
Causal Links = {putdown(a) → on(table(a) → Finish,  
                unstack(a,b) → holding(a) → putdown(a),  
                Start → handempty → unstack(a,b),  
                putdown(c) → on(c, table) → Finish,  
                unstack(c,d) → holding(c) → putdown(c),  
                Start → handempty → unstack(c,d),  
                Start → block(c) → unstack(c,d),  
                Start → block(d) → unstack(c,d),  
                Start → on(c,d) → unstack(c,d)}
```

```
Open preconditions = {}
```

Die Aktion `unstack(a,b)` verursacht ein Konflikt mit dem kausalen Link `Start → handempty → unstack(c,d)`, da es `¬handempty` als Effekt hat und zwischen `Start` und `unstack(c,d)` eintreten kann. Dieser Konflikt kann nur dadurch gelöst werden, in dem das Ordering `unstack(c,d) < unstack(a,b)` eingefügt wird. Das allerdings wäre ein Widerspruch zu dem bereits eingefügten Ordering `unstack(a,b)`

< unstack(c, d). Damit müssen wir ein Backtracking zu (*) machen und eine andere Aktion, nämlich putdown(a), auswählen:

```
Actions = {Start, Finish, putdown(a), unstack(a, b), putdown(c), unstack(c, d) }
Orderings = {Start < Finish,
             Start < putdown(a), putdown(a) < Finish,
             Start < unstack(a, b), unstack(a, b) < Finish,
             unstack(a, b) < putdown(a),
             Start < putdown(c), putdown(c) < Finish,
             Start < unstack(c, d), unstack(c, d) < Finish,
             unstack(a, b) < unstack(c, d)}
             putdown(a) < unstack(c, d)}
Causal Links = {putdown(a) → on(table(a) → Finish,
                unstack(a, b) → holding(a) → putdown(a),
                Start → handempty → unstack(a, b),
                putdown(c) → on(c, table) → Finish,
                unstack(c, d) → holding(c) → putdown(c),
                putdown(a) → handempty → unstack(c, d)}
Open preconditions = {block(c), block(d), on(c, d)}
```

Refining - folgende mögliche Aktionen stehen zur Auswahl:

Start

Es wird Start gewählt. Das geschieht nun drei Mal, je für block(c), block(d) und on(c, d):

Actions = {Start, Finish, putdown(a), unstack(a,b), putdown(c), unstack(c,d) }

Orderings = {Start < Finish,
Start < putdown(a), putdown(a) < Finish,
Start < unstack(a,b), unstack(a,b) < Finish,
unstack(a,b) < putdown(a),
Start < putdown(c), putdown(c) < Finish,
Start < unstack(c,d), unstack(c,d) < Finish,
unstack(a,b) < unstack(c,d)}
putdown(a) < unstack(c,d)}

Causal Links = {putdown(a) → on(table(a) → Finish,
unstack(a,b) → holding(a) → putdown(a),
Start → handempty → unstack(a,b),
putdown(c) → on(c, table) → Finish,
unstack(c,d) → holding(c) → putdown(c),
putdown(a) → handempty → unstack(c,d),
Start → block(c) → unstack(c,d),
Start → block(d) → unstack(c,d),
Start → on(c,d) → unstack(c,d)}

Open preconditions = {}

Da nun keine Preconditions mehr offen sind, sind wir fertig.

Unten sieht man eine Version der Lösung, in der redundante Ordering Bedingungen entfernt wurden.

```
Actions = {Start, Finish, putdown(a), unstack(a,b), putdown(c), unstack(c,d) }
Orderings = {Start < unstack(a,b), unstack(a,b) < putdown(a),
             unstack(c,d) < putdown(c), putdown(c) < Finish,
             unstack(a,b) < unstack(c,d), putdown(a) < unstack(c,d)}
Causal Links = {putdown(a) → on(table(a) → Finish,
                unstack(a,b) → holding(a) → putdown(a),
                Start → handempty → unstack(a,b),
                putdown(c) → on(c, table) → Finish,
                unstack(c,d) → holding(c) → putdown(c),
                putdown(a) → handempty → unstack(c,d),
                Start → block(c) → unstack(c,d),
                Start → block(d) → unstack(c,d),
                Start → on(c,d) → unstack(c,d)}
Open preconditions = {}
```

b) Der oben gefundene Plan ist linear, und erlaubt daher nur eine Abarbeitung.