Gliederung

1. KI im Allgemeinen und in dieser Vorlesung
2. Heuristische Suche
3. Logik und Inferenz
4. Wissensrepräsentation
5. **Handlungsplanung**
6. Lernen
7. Sprachverarbeitung
8. Umgebungswahrnehmung

1. Grundlagen
2. „Neoklassisches“ Planen
3. Planen unter Unsicherheit

Weiterführendes Material zum Planen

- M. Ghallab, D. Nau. P. Traverso: *Automated Planning – Theory and Practice*. Morgan Kaufmann, 2004
- http://www.planet-noe.org/service/index.html
  -> Repositories & Research Sites
  Umfangreiche Links zu aktuellen Planungssystemen und Benchmark-Problemen

Planen und Suche

Ein **Plan** ist eine Struktur, die Repräsentationen von Handlungen und Zielen enthält und dazu dient, über die Wirkung zukünftiger Handlungen zu räsonieren und die zielgerichtete Ausführung von Handlungen zu beeinflussen.

Pläne sind schon früher in der Vorlesung vorgekommen:

- Verschiebespiel, Reiserouten in Rumänien, Soko-Ban, …

↪ Warum nun noch ein eigenes Kapitel dazu?

Planung ist wie Suche mit besonderer Struktur

	Suche	**Planung**
Zustände	Datenstrukturen	Logik-Ausdrücke
Aktionen	Code	Vor-, Nachbedingung
Ziel	Code	Logik-Ausdrücke
Plan	Akt.-Folge von *Start*	Constraints auf Akt.n

Varianten von Planung

Es gibt unterschiedliche Anforderungen an Planung und Pläne:

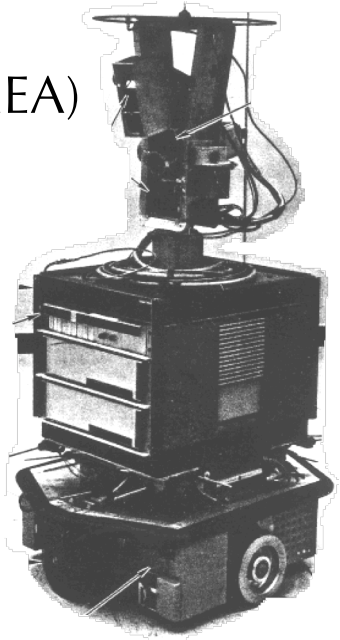
- **Zeitmodell**: Situationen/Intervalle, qualitativ/quantitativ,
- **Aktionen**: Kontextfrei/kontextabhängig, deterministisch/nicht-deterministisch
- Vollständige/unvollständige **Bereichsinformation** bei Planung oder Ausführung
- **Anwendungsbereich** dynamisch/statisch bei Planung oder Ausführung
- ein/mehrere **Planer**, ein/mehrere **Ausführer**
- **Plan„bibliothek“** vorhanden oder nicht
- **Planung** korrekt & vollständig & optimal oder nicht, ressourcenabhängig oder nicht

Die KI hat Arbeiten zu sehr vielen dieser Varianten;
in dieser Vorlesung werden wir nur sehr wenige davon streifen

Wie alles begann

Handlungsplanung stand von Beginn im Zentrum der KI, damals als Teil von „Problemlösen“ (*problem solving*) gewertet

- **GPS** *(General Problem Solver)*, Newell/Shaw/Simon,
  Ende 1950er bis Ende 1960er Jahre:
  Allgemeines Modell und Implementierung menschlicher Problemlösung
  Zielrichtung: KI in Richtung Kognitive Psychologie
  Einfluss auf Planung: Rückwärtssuche (*Means-Ends Analysis,* MEA)
- **STRIPS** (*Stanford Research Inst. Problem Solver*),
  Fikes/Hart/Nilsson,
  Ende 1960er bis Mitte 1970er Jahre
  Planungs-/Ausführungskomponente des Roboters SHAKEY
  Zielrichtung: Algorithmische, Logik-orientierte, System-KI
  extrem einflussreich auf die KI über Jahrzehnte (A*, Robotik)
  Einfluss auf Planung: Repräsentation,
  Planungsalgorithmus (ausgehend von MEA)



SHAKEY, 1969

Propositionales Planen

Einfachste Planungsvariante, „klassisches“ Planen
(manchmal auch STRIPS-Planen genannt, obwohl STRIPS etwas Anderes gemacht hat, teils mehr, teils weniger)

Voraussetzungen/Annahmen

- Zeitmodell: qual. Situationen,
- Aktionen: K.frei, deterministisch,
- vollständige Bereichsinformation
- Anwendungsbereich statisch
- ein Planer,
- keine Plan„bibliothek“

Gegeben Problemrepräsentation:

- Startsituation („viele Fakten“)
- Zielbedingungen („wenige Fakten“)
- Operatorschemata
  - Vorbedingungen („wenige Fakten“)
  - Nachbedingungen („wenige Fakten“)

Finde Lösungsplan:
Partiell geordnete Menge von Operatorinstanzen, wohlgeformt und konfliktfrei

Basisdefinitionen

Ein **Plan** ist ein Paar $\langle O, < \rangle$ aus einer Menge O von Operatorinstanzen und einer Ordnungsrelation $<$ auf O.

Ein **Operator** ist ein Paar $\langle V, N \rangle$ aus Mengen V, N von Grundfakten. V sind die **Vorbedingungen** des Operators (was muss gelten, damit er ausgeführt werden kann), N sind seine **Nachbedingungen** (was sind die Effekte seiner Ausführung)

Bei einem **STRIPS-Operator** ist N aufgeteilt in D, W: die Fakten, die positiv dazu kommen, und die, die wegfallen.

Sei S eine **Situation**, d.h. eine Menge von Grund-Fakten.
Das **Resultat** der **Anwendung** eines STRIPS-Operators $o=\langle V,D,W \rangle$ in S ist

$$S' = \begin{cases} S & \text{if} \quad V \not\subseteq S \\ (S \setminus W) \cup D & \text{else} \end{cases}$$

Löst das Frame-Problem!

Demo-Beispiel: Die Blockwelt

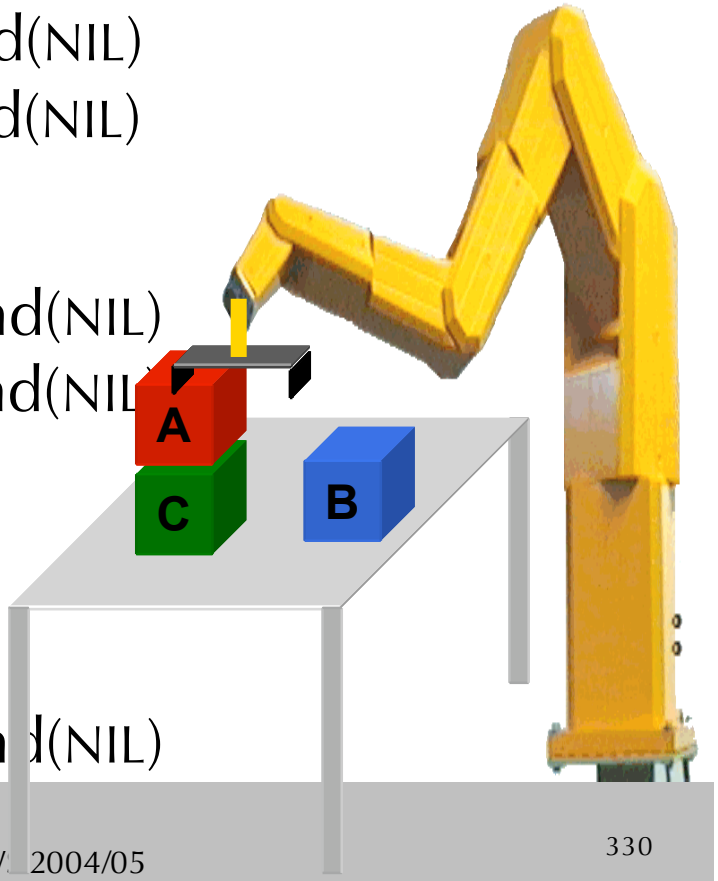
Stack(x,y): **Vor**: Hand(x), Free(y)
Weg: Hand(x), Free(y)
Dazu: On(x,y), Free(x), Hand(NIL)

Unstack(x,y): **Vor**: On(x,y), Free(x), Hand(NIL)
Weg: On(x,y), Free(x), Hand(NIL)
Dazu: Hand(x), Free(y)

Pick(x): **Vor**: Ontbl(x), Free(x), Hand(NIL)
Weg: Ontbl(x), Free(x), Hand(NIL)
Dazu: Hand(x)

Put(x): **Vor**: Hand(x)
Weg: Hand(x)
Dazu: Ontbl(x), Free(x), Hand(NIL)

STRIPS-Operator-Schemata



Domänenbeschreibungen

Um eine **Planungsdomäne** (wie Blockwelt) zu beschreiben, müssen Prädikate, Typen, Objekte, Operatoren definiert werden!
PDDL *(Problem Domain Description Language)* erlaubt Varianten, z.B.:

STRIPS-Sprache	**ADL-Sprache**
Situation = Menge pos. Grundfakten {Poor, Unknown}	Situation = Menge v. Literalen {Poor, ¬Famous}
CWA: Fakt nicht erwähnt ➥ falsch	Fakt nicht erwähnt ➥ unbestimmt
Ziel = Menge pos. Grundfakten Menge = Konjunktion {Rich, Famous}	Ziele dürfen quantifizieren, Disjunktion erlaubt ∃x.[On(A,x) ∨ On(B,x)]
Keine Sortierung v. Objekten	Sortierung, zB (A,B,C,NIL – Block)

ADL = *Action Definition Language*
STRIPS, ADL haben weitere Unterschiede; es gibt weitere Varianten in PDDL

Problembeschreibungen

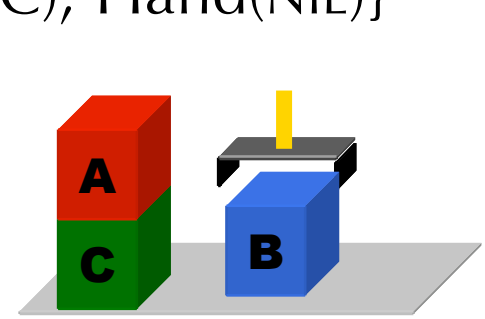
Innerhalb einer Planungs*domäne* gibt es Planungs*probleme*

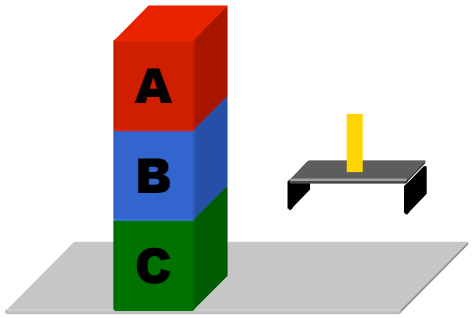
Eine **STRIPS-Problembeschreibung** ist ein Tripel $\langle S,O,F\rangle$ aus
S: Menge von Grund-Fakten (Startsituation)
O: Menge von STRIPS-Operatoren (bzw. Schemata)
F: Menge von Grund-Fakten (Zielbedingungen)

Beispiel (Operatorschemata, Objekte wie zuvor)

{Ontbl(C), Ontbl(B), Free(B),
Free(A), On(A,C), Hand(NIL)}

{On(A,B), On(B,C)}

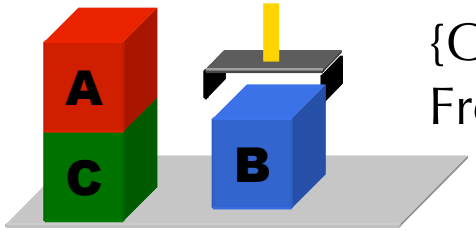




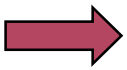
Lösungen

Gegeben eine (STRIPS-)Problembeschreibung.
Ein Plan $\langle O, < \rangle$ ist eine **Lösung** des Problems, wenn Ausführung aller Operatoren aus O in einer mit $<$ verträglichen Reihenfolge in einer Situation resultiert, welche die Zielbedingungen erfüllt.

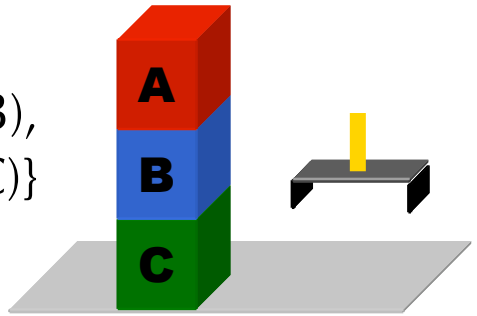
Beispiel Der folgende Plan (Op.en in der genannten Ordnung) ist eine Lösung des Problems



{Ontbl(C), Ontbl(B), Free(B), Free(A), On(A,C), Hand(NIL)}



{On(A,B), On(B,C)}

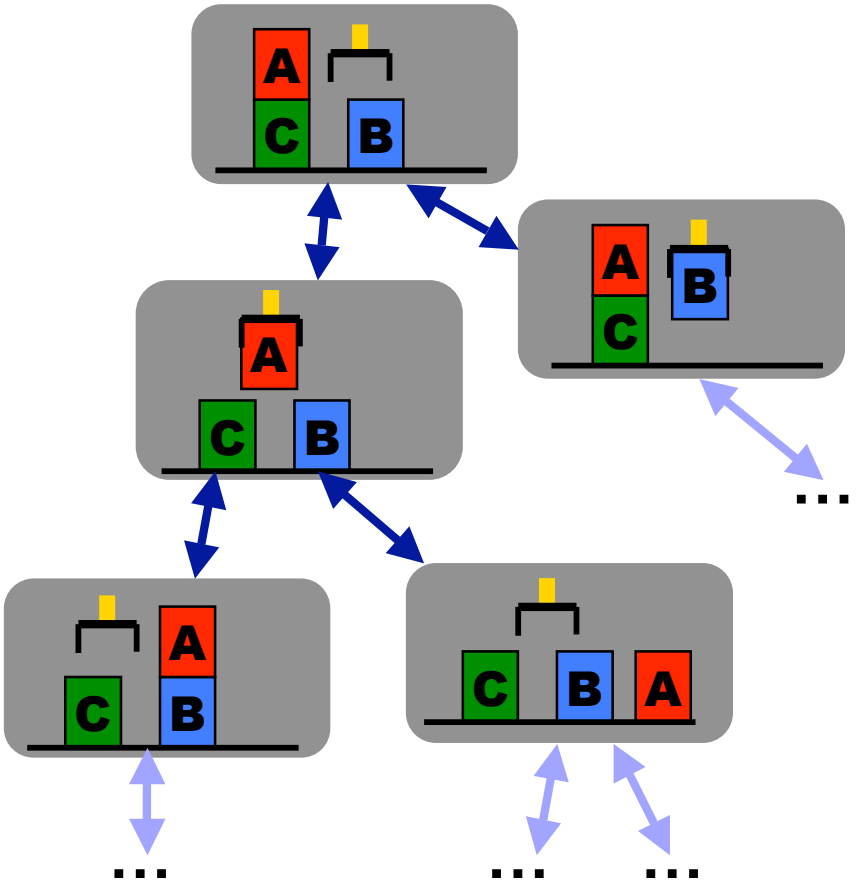


⟨**Unstack**(A,C), **Put**(A), **Pick**(B), **Stack**(B,C), **Pick**(A), **Stack**(A,B)⟩

Suchräume zur Planung

Situationenraum

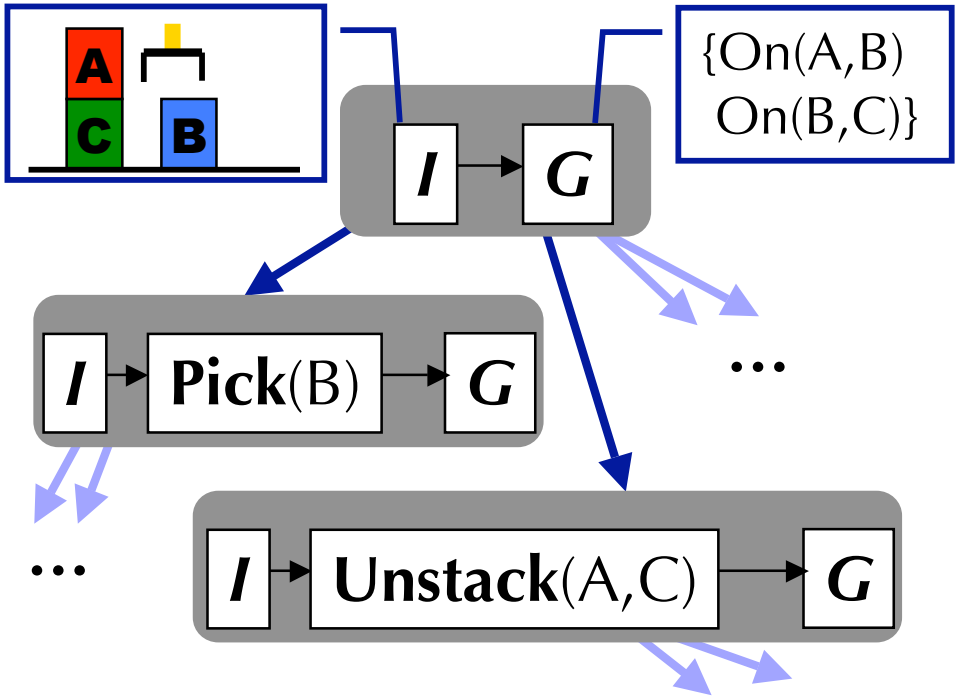
Knoten: Situationen



vgl. Suchraum Kap. *Suche*

Planraum

Knoten: (meist unfertige) Pläne



Pseudo-Operatoren *I*, *G*

I: Start-Sit. als Nachbedingung

G: Ziele als Vorbedingung

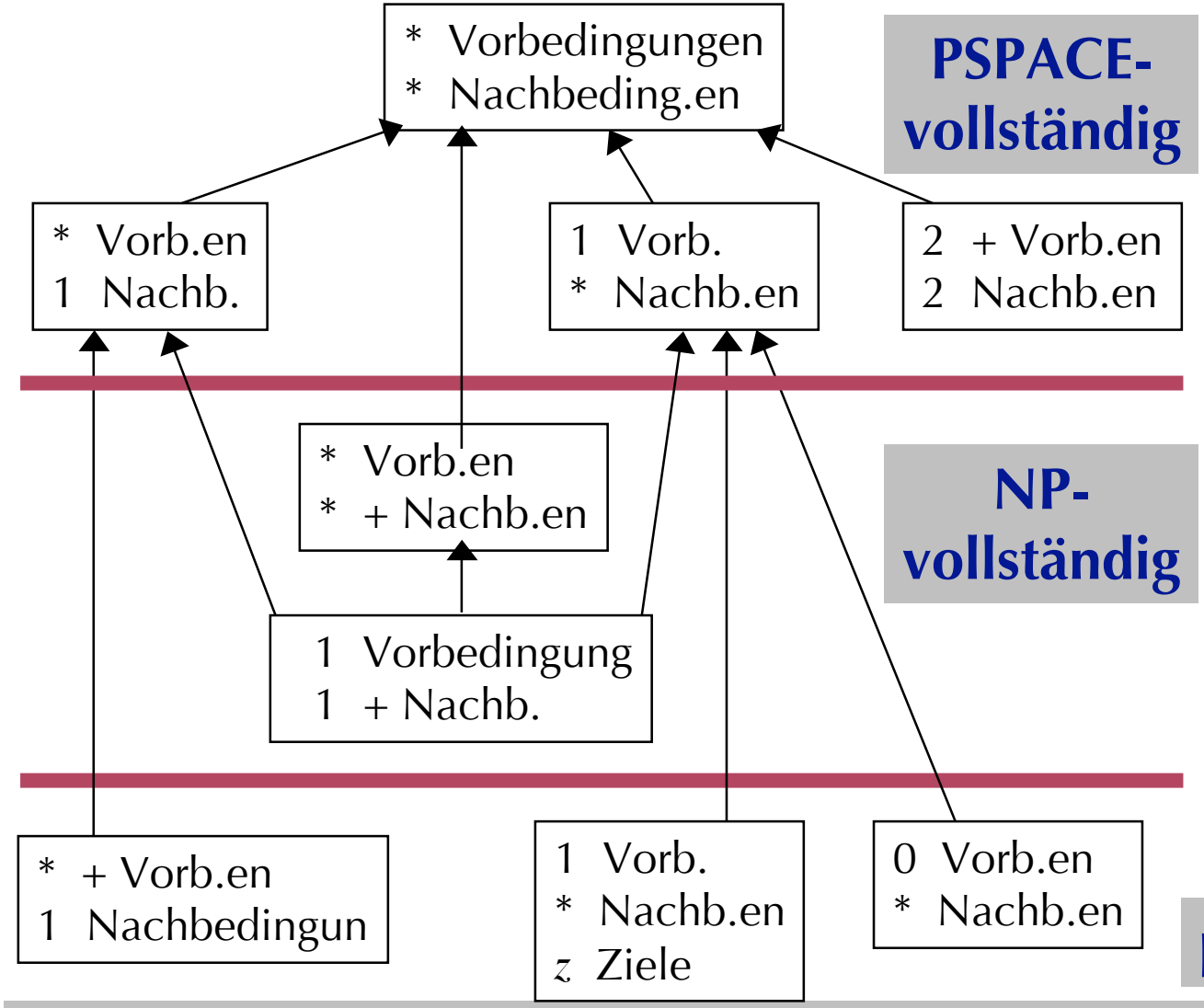
Progression und Regression

Suchrichtung rückwärts im Situationenraum (*means-ends analysis*) fokussiert automatisch auf zielrelevante Operatoren,
z.B. Blockwelt mit 1000 Blöcken und Ziel {On(A,B), On(B,C)}

Regression

- Soll nach Operator o eine seiner Nachbedingungen $n(o)$ gelten, so müssen vorher seine Vorbedingungen $V(o)$ gelten (und $\neg n(o)$).
  ➥ $V(o)$ ist Ergebnis der Regression von $n(o)$ über o.
- Entsprechend Regression von Merkmal*mengen* über Operator:
  z.B. {On(A,B), On(B,C)} regrediert über **Stack**(A,B) ergibt {On(B,C), Hand(A), Free(B)} (+ ggf. Negation der Vorbedingungen)
- Komplexere Beschreibungssprachen führen zu komplexeren Regressionsergebnissen (z.B. variable Operatorargumente)
- Die andere Richtung (wie Vorwärtssuche): **Progression**

Die Komplexität des PLANSAT-Problems



PSPACE-vollständig

NP-vollständig

PLANSAT

Gegeben propositionales Problem, gibt es eine Lösung?

[Bylander, 1994]

Realistische Probleme machen exponenziellen Aufwand!

polynomiell

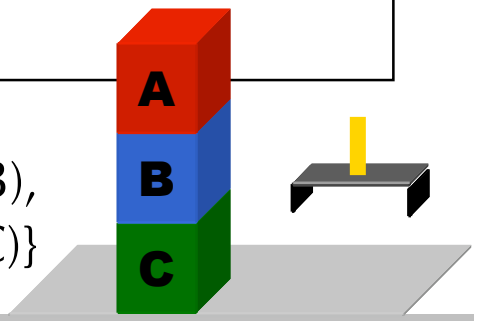
Der STRIPS-Algorithmus, nichtdet. Form

function STRIPS-PLANNER($\langle S,O,F\rangle$) **returns** a plan (operator sequence)
input: $\langle S,O,F\rangle$, a STRIPS problem representation
local variables: P, P': Plan (operator sequence)
begin
1. $P \leftarrow \lambda$ (empty sequence)
2. **while** still goals of F open in S **do**
2.1 **choose** goal $f \in F$ open in S
2.2 **choose** operator instance $o \in O$ adding f ;
return(**failure**) if no such o exists
2.3 $P' \leftarrow$ STRIPS-PLANNER($\langle S,O, o.preconditions\rangle$);
return(**failure**) if P'=**fail**
2.4 $S \leftarrow [P' \oplus o](S)$; $P \leftarrow P \oplus P' \oplus o$ ($\oplus$ = concatenation)
return(P) **end**

Wahlen in Schritt 2.1, 2.2 sind kritisch!

z.B. wähle Ziel On(B,C) in Demo-Problem zuerst!

{On(A,B), On(B,C)}



Heuristiken für STRIPS-PLANNER

- Planungsalgorithmen induzieren *zwei Kostenarten*:
  - Kosten der *Planung* (= Suchkosten, Kapitel Suche)
  - Kosten des *Plans* (z.B. Operatoren x Operatorkosten)
- Plankosten hier vereinfacht: Anzahl Operatoren
  (also: Bevorzuge „kürzere“ Pläne)
- zulässige Heuristiken (-> A*-Suche):
  - Unabhängigkeit von Teilzielen
  - Vernachlässigen aller Vorbedingungen
  - Vernachlässigen der „Weg“-Bedingungen