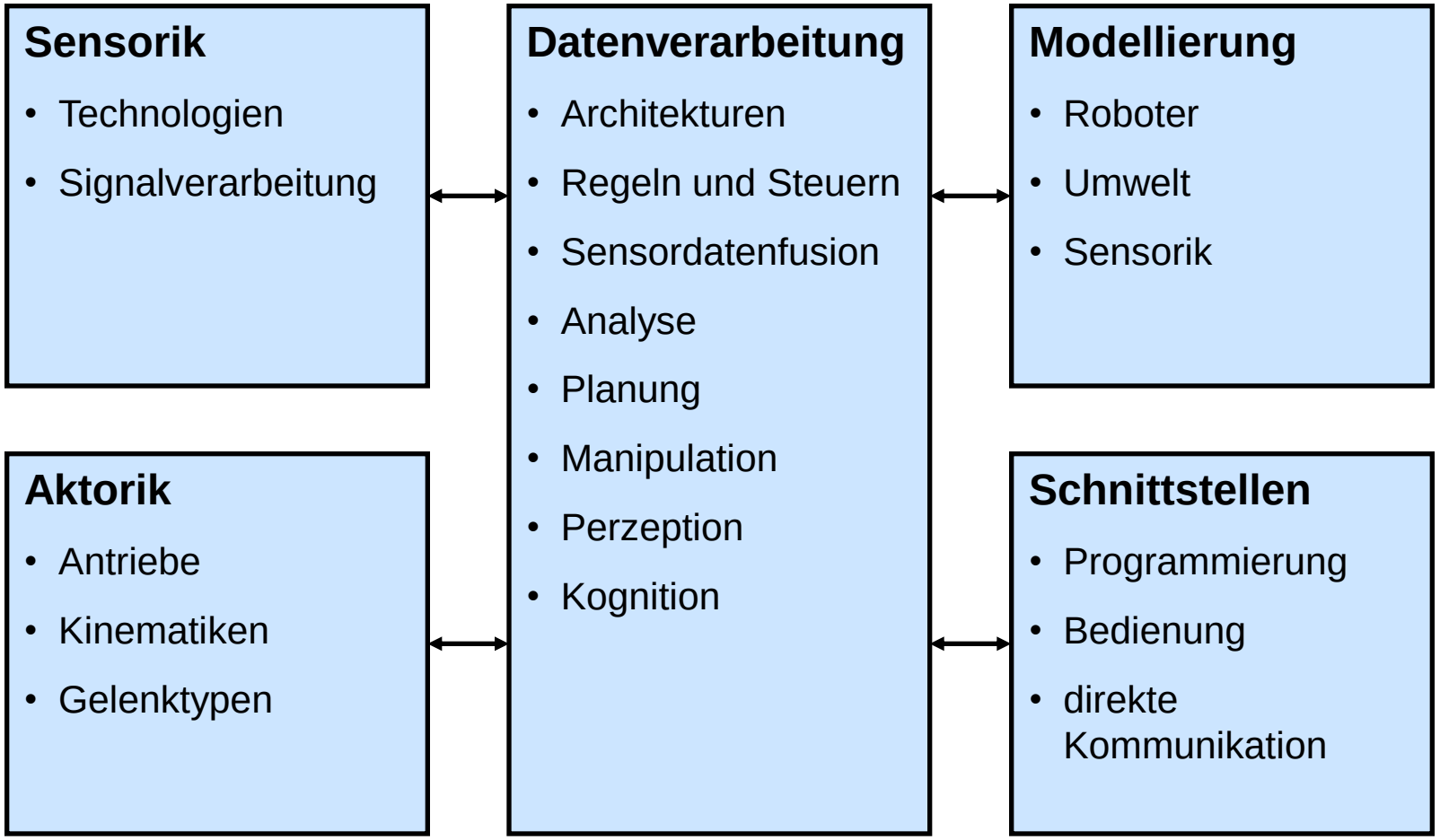
XII. Planungssysteme

Prof. Dr.-Ing. Rüdiger Dillmann

Teilbereiche



Inhalt

- Motivation
- Planungsverfahren
- Planung in der Robotik
- STRIPS

Motivation

Warum planen?

Was planen?

Wie planen?

Motivation

Warum planen?

- Planung dort, wo mehr als ein Schritt nötig ist, um zum Ziel zu kommen (eine Aufgabe zu erledigen).

Motivation

Was planen?

- Aktionsplanung
  - Welche Aktionen werden aus Aufgabe abgeleitet?
- Reihenfolgeplanung
  - Zeitliche Beziehungen zwischen Aktionen
- Ressourcenplanung
  - Welche Roboter, Geräte führen Aktionen aus
- Zeitplanung
  - Wann werden Aktionen gestartet
- Durchführungsplanung
  - Welche Parameter für Aktionsausführung (Bahn- und Greifplanung)
- Scheduling
  - Ressourcenplanung und Zeitplanung

Motivation

Wie planen?

Unter einem **Planungssystem** für Roboter versteht man allgemein ein System, das ausgehend von einem **Anfangszustand** und der Beschreibung eines gewünschten **Zielzustandes** eine Folge von **Aktionen** (Aktionsplan) generiert, die das betrachtete System von seinem Anfangszustand schrittweise in den gewünschten Zielzustand überführen.

- Überwachung des Plans mit Sensoren
- bei Abweichung umplanen

Inhalt

- Motivation
- Planungsverfahren
  - Vorgehensweise
  - Suchverfahren
- Planung in der Robotik
- STRIPS

Planungsverfahren

Definitionen

- Die von einem Planer erzeugten Aktionssequenzen oder Graphen werden als **Plan** bezeichnet.
- Der Plan beschreibt im Zustandsraum eine Menge von **Zustandsänderungen**, die zum gewünschten Zielzustand führen.
- Diese Zustandsänderungen können **sequentiell, nebenläufig oder koordiniert nebenläufig** sein.
- Generell können Pläne in Teilpläne zerlegt werden
  - Plansegmente: vollständige Teilpläne
  - Planskelette: unvollständige Teilpläne
- Ein nicht weiter zerlegbares Plansegment bzw. der Zustandsübergang wird als Aktionsprimitiv oder **Planprimitiv** bezeichnet.

Planungsverfahren

Vorgehensweise

1. Ziel-Formulierung

- Gesucht: klare Definition, woraus das Ziel besteht
- Definition „Ziel“: Exakt die Menge von Weltzuständen, die das Ziel erfüllen

2. Problem-Formulierung

- Gesucht: Zustände und Aktionen, um zum Ziel zu kommen
- Frage der Abstraktion

Planungsverfahren

Vorgehensweise

Hintergrundwissen

- Ohne Hintergrundwissen keine Planung möglich
- In Form von Karten, Handlungswissen, ...
- Verschiedene Sequenzen von Aktionen erzeugen:
  - Mögliche, gültige Sequenzen ermitteln
  - „Beste“ Sequenz auswählen

3. Suche

4. Ausführung

Planungsverfahren

Genereller Ablauf

```
while (true) do
    Zustand := UPDATESTATE(Zustand, Perzeption)
    Ziel := FORMULATEGOAL(Zustand)
    Problem := FORMULATEPROBLEM(Zustand, Ziel)
    Sequenz := SOLVEPROBLEMBYSEARCH(Problem)
    EXECUTE(Sequenz)
end while
```

Planungsverfahren

Formale Definition von Problemen und Lösungen

- **Initialer Zustand s^0**
- **Nachfolgerfunktion,** Liste möglicher Nachfolgezustände und Aktionen aus Zustand s^t

  SUCC-FN(s^t) = {(a^{t+1},s^{t+1}); s^t geht in s^{t+1} über mit Aktion a^{t+1}}
- **Zustandsraum**, als Graph:
  - Knoten: Zustände
  - Kanten: Zustandsübergänge, Aktionen
- **Pfad**
  - Folge von Zuständen, verknüpft durch Aktionen

Planungsverfahren

Formale Definition von Problemen und Lösungen

- **Zieltest**
  - Menge aller Zielzustände
  - Funktion (z.B. „Schachmatt“)
- **Pfadkosten**
  - Zeit, Wegstrecke, Kosten, ...
- **Schrittkosten**
  - Kosten entlang einer Kante
- **Lösung**
  - Pfad vom Startzustand zu einem Zielzustand
  - Optimal: geringste Pfadkosten

Planungsverfahren

Beispiele für Probleme

- 8-Damen Problem
- TSP (Travelling Salesman Problem)
- Türme von Hanoi

- VLSI-Layout
- Navigation für Roboter
- Automatic Assembly Sequencing

Planungsverfahren

Unterschiedliche Problemstellungen

Vollständiges Wissen	↔	Unvollständiges Wissen
Determinismus	↔	Nichtdeterminismus
Endlicher Zustandsraum	↔	Unendlicher Zustandsraum
Diskreter Zustandsraum	↔	Kontinuierlicher Zustandsraum
Statischer Zustandsraum	↔	Dynamischer Zustandsraum

Inhalt

- Motivation
- Planungsverfahren
  - Vorgehensweise
  - Suchverfahren
- Planung in der Robotik
- STRIPS

Suchverfahren

Problemstellung

Gegeben:

- Problem-Formulierung (Zustandsraum als Graph od. SUCC-FN)
- Ziel-Formulierung (Zielzustände od. Zieltest-Funktion)
- Initialer Zustand

Gesucht:

Pfad vom initialen Zustand zum Ziel unter bestimmten Nebenbedingungen (z.B. minimale Kosten).

➔ Suche im Zustandsraum notwendig

Suchverfahren

Allgemeines Verfahren (I)

Baumsuche, Wurzelknoten ist Startzustand

Ablauf:

1. Vergleiche aktuellen Zustand mit Zielzustand
2. Expandiere aktuellen Zustand durch Anwenden der Nachfolgerfunktion SUCC-FN(x)
   - Erzeugt neue Menge von Zuständen
3. Wähle einen der neuen Zustände aus (merke alte) und gehe zu 1

Bemerkungen:

- Suchbaum != Zustandsraum
- Knoten im Baum != Zustand
- Speichern der erzeugten Knoten als Liste

Suchverfahren

Allgemeines Verfahren (II)

```
FUNCTION BaumSuche(Problem) RETURN Lösung oder Fehler

    Liste.Insert(Startzustand)
    WHILE (true) DO
        IF(Liste.IsEmpty())
                THEN RETURN Fehler
        Knoten = Liste.FirstElement()
        IF (ZielTest(Knoten, Problem))
                THEN RETURN Solution(Knoten)
        Liste.Insert(Expand(Knoten, Problem))
```

Suchverfahren

Allgemeines Verfahren (III)

```
FUNCTION Expand(Knoten,Problem) RETURN Liste von Knoten

    Rückgabeliste = Leere Liste
    FOR EACH <Aktion, Ergebnis> IN SUCC-FN(Knoten.Zustand) DO
        Knoten s = neuer Knoten
        s.Zustand = Ergebnis
        s.Elternknoten = Knoten
        s.Aktion = Aktion
        s.Pfadkosten = Knoten.Pfadkosten + Schrittkosten(Knoten, Aktion, s)
        s.Tiefe = Knoten.Tiefe + 1
        Rückgabeliste.Add(s)

    RETURN Rückgabeliste
```

Suchverfahren

Bewertungskriterien

- Vollständigkeit
  - Wird Lösung gefunden, falls eine existiert?
- Optimalität
  - Wird optimale Lösung gefunden?
- Zeitkomplexität
- Raumkomplexität, Speicheraufwand

Komplexitätsmaße:

- b: Verzweigungsfaktor (branching factor)
- d: Tiefe der Lösung (depth)
- m: maximale Tiefe des Baums

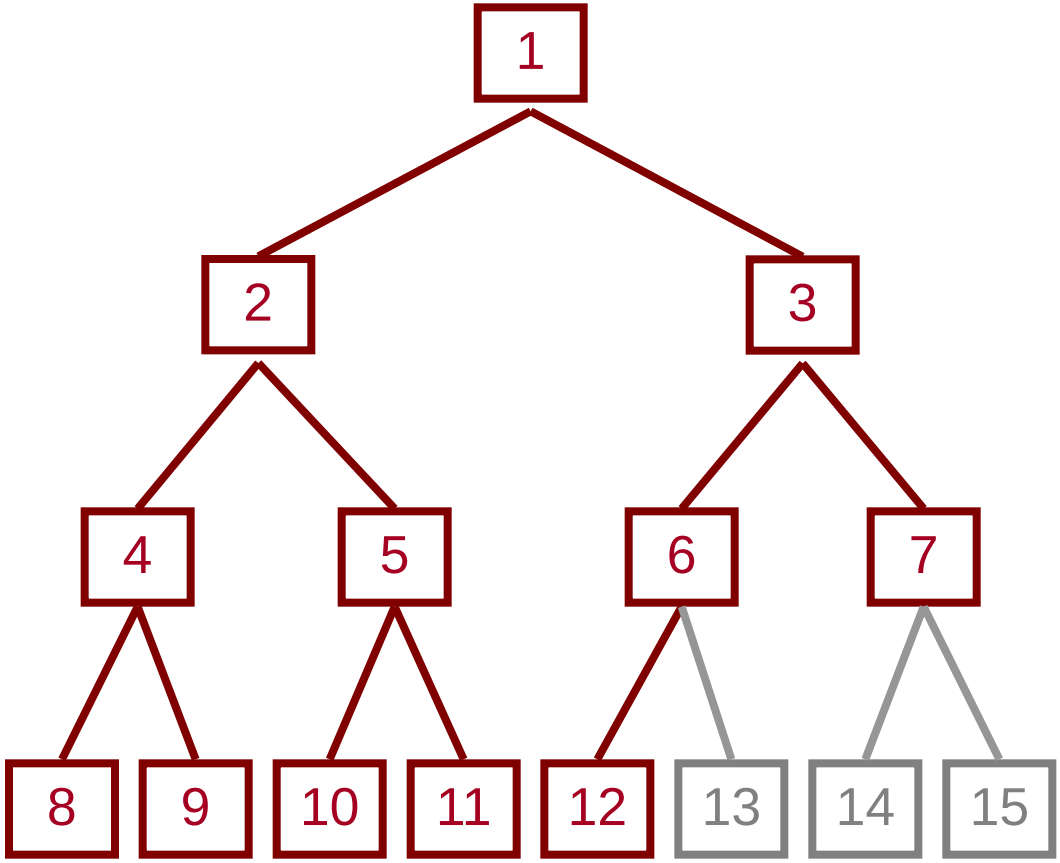
Suchverfahren

Übersicht

- Analytische Verfahren
  - Breitensuche
  - Tiefensuche
  - Iterative Deepening
  - Bidirektionale Suche
- Heuristische Verfahren
  - Dekomposition
  - A*

Suchverfahren

Breitensuche (I)



- Beginn am Ausgangszustand
- Prüfe alle Knoten gleicher Tiefe
- Gehe dann eine Stufe tiefer

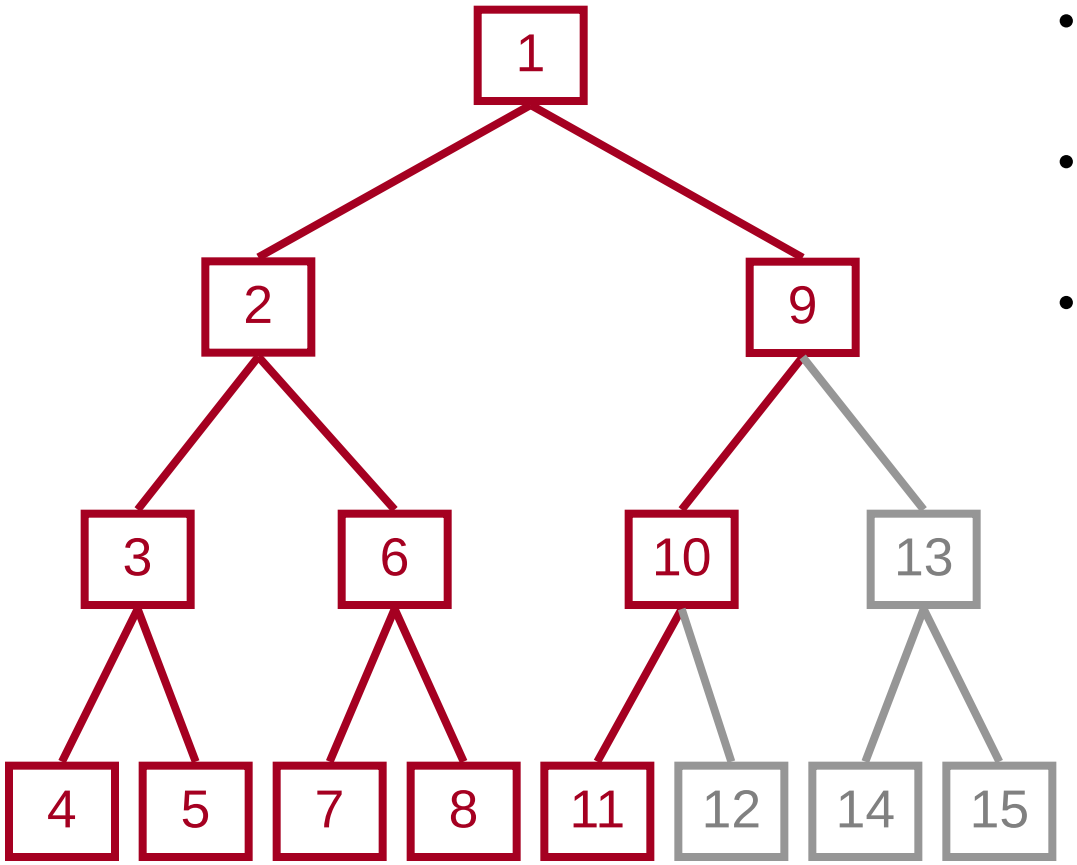
Breitensuche (II)

- Implementierung:
  - Liste als FIFO (neue Elemente hinten anhängen)

- Bewertung:
  - Vollständig
  - Optimal (falls alle Kantengewichte identisch)
  - Aufwand sehr hoch: $O(b^{d+1})$

Suchverfahren

Tiefensuche (I)



- Beginn am Ausgangszustand
- Suche bis zur maximalen Tiefe
- Weiter im tiefsten Knoten, der noch nicht-durchsuchte Kanten hat

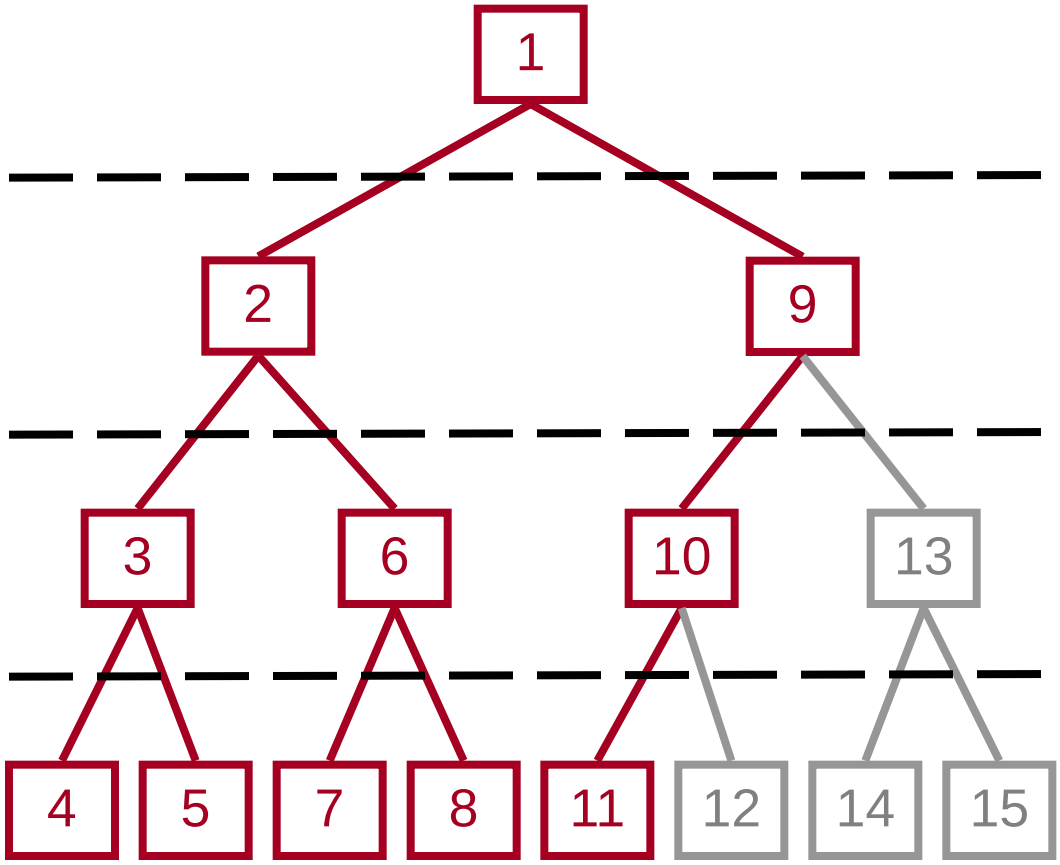
Tiefensuche (II)

- Implementierung:
  - Liste als Stack (LIFO) (neueste Elemente vorne anhängen)

- Bewertung:
  - Vollständig? Ja und Nein (Zyklen).
  - Optimalität? Nein.
  - Aufwand:
    - Geringer Speicherbedarf
    - Hohe Laufzeit

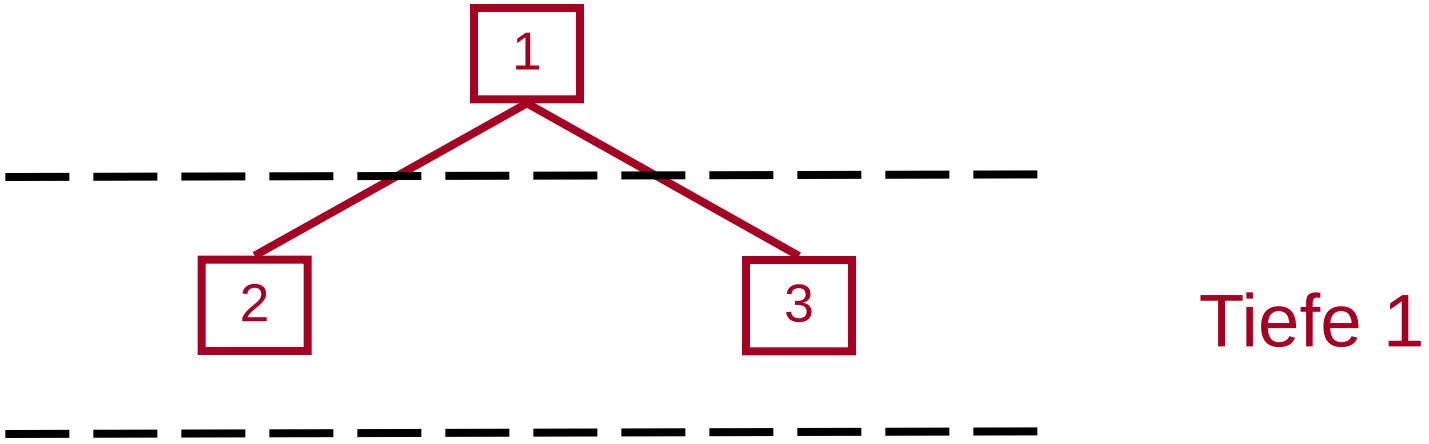
Suchverfahren

Iterative Deepening (I)

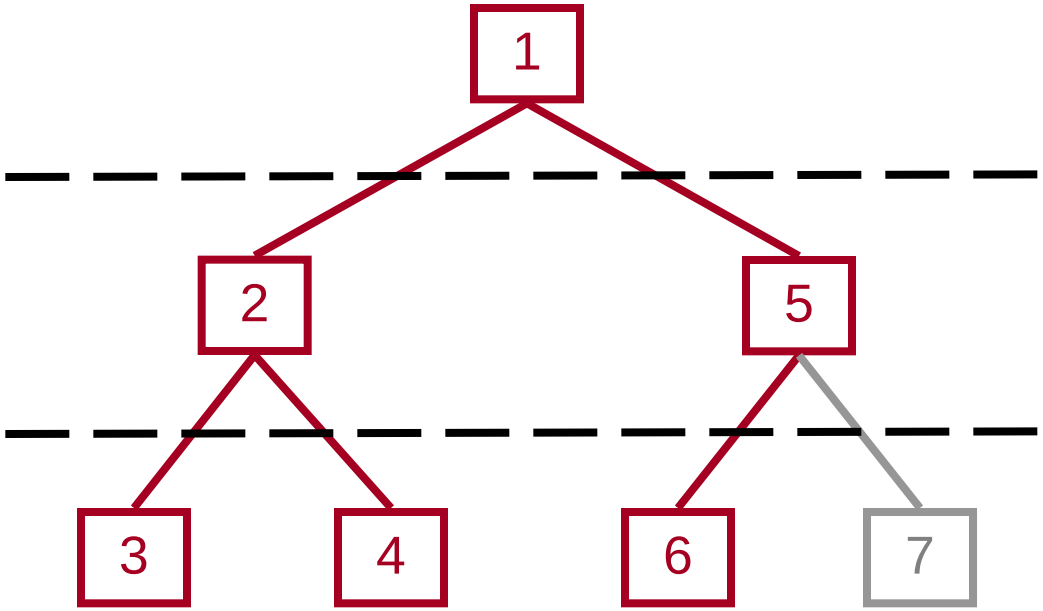


- Beginn am Ausgangszustand
- Tiefensuche bis zur Tiefengrenze
- Tiefengrenze inkrementieren, falls keine Lösung gefunden

Iterative Deepening (II)



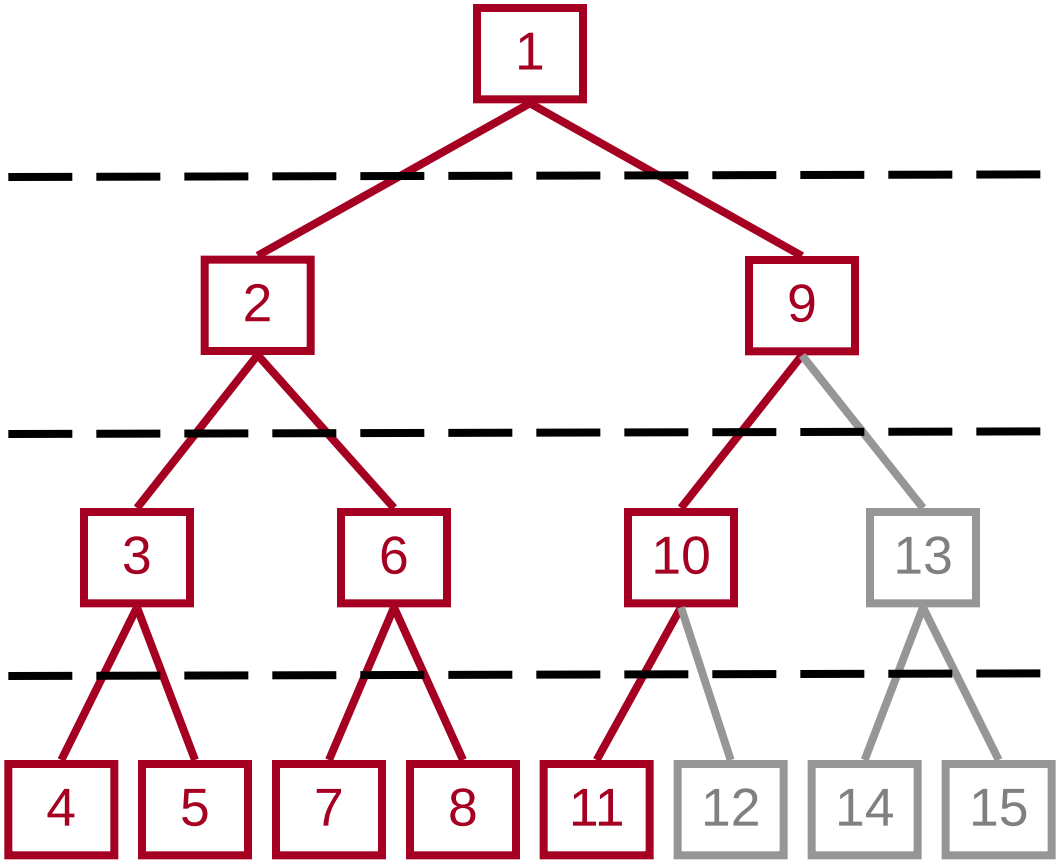
Iterative Deepening (II)



Tiefe 2

Suchverfahren

Iterative Deepening (II)



Tiefe 3

Suchverfahren

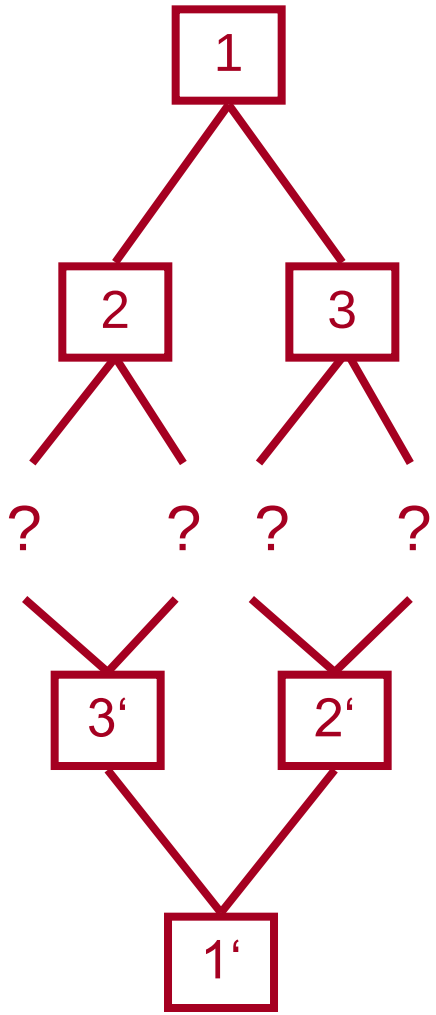
Iterative Deepening (III)

- Erweiterung der Tiefensuche um maximale Suchtiefe

- Bewertung:
  - Methode der Wahl, wenn Suchraum groß und Tiefe a priori unbekannt

Suchverfahren

Bidirektionale Suche



- Beginne eine Breitensuche am Ausgangszustand und eine weitere am gewünschten Zielzustand
- Halte, wenn bei beiden Suchen ein identischer Zustand erreicht wurde
- die Lösung setzt sich zusammen aus den beiden Pfaden, mit denen dieser identischer Zustand erreicht wurde
- Problem: Existenz einer Vorgängerfunktion (PRED-FN)

Suchverfahren

Vergleich der analytischen Suchverfahren

	Breitensuche	Tiefensuche	Iterative Deepening	Bidirektionale Suche
Vollständigkeit	Ja	Nein	Ja	Ja
Laufzeit	$O(b^{d+1})$	$O(b^{m})$	$O(b^{d})$	$O(b^{d/2})$
Speicherbedarf	$O(b^{d+1})$	$O(bm)$	$O(bd)$	$O(b^{d/2})$
Optimal	Ja*	Nein	Ja*	-

- Legende:
  - b: Verzweigungsgrad
  - d: Tiefe der günstigsten Lösung
  - m: max. Tiefe des Suchbaums

* Falls Pfadkosten identisch in allen Kanten

Aus: Russel & Norvig:
Artificial Intelligence – A Modern Approach, 2nd Edition

Suchverfahren

Heuristiken

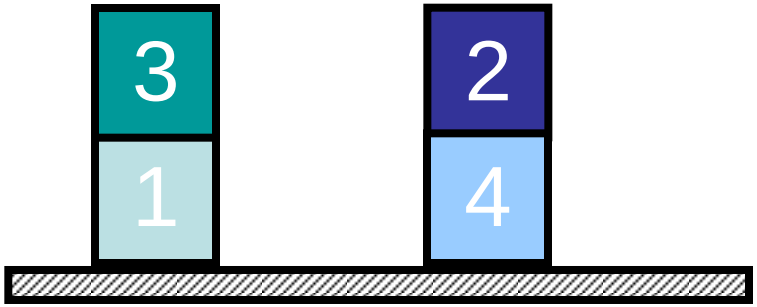
Möglich, wenn zusätzliches Wissen zugänglich ist, z.B.

- vollkommene Unabhängigkeit bestimmter Teilaufgaben (Dekomposition der Gesamtaufgabe)
- Schätzung der ‚Distanz' zwischen einem Zwischenzustand und dem gewünschten Endzustand (A*-Suche)
- Ignoranz irrelevanter Informationen und Ausschluss von Aktionen, die keine Annäherung zum Zielzustand bringen

Suchverfahren

Dekomposition

- Hier: Zwei unabhängig voneinander erreichbare Teilaufgaben
- Lösung:
  - Finde einen Plan für jede Teillösung
- Probleme:
  - Teillösungen können sich gegenseitig blockieren oder aufheben



Gewünschter Endzustand

Suchverfahren

A*-Verfahren (I)

Eigenschaften

- Baumsuche
- Suche nicht systematisch entlang der Baumstruktur
- Heuristik h(n)
  - Funktion, die die ‚Distanz' (Kosten) des Zustandes n zum Zielzustand schätzt
- Funktion g(n)
  - Ermittelt die tatsächlichen Kosten vom Ausgangszustand zum Zustand n
- Suche jeweils in dem Knoten fortsetzen, in dem f(n) = g(n) + h(n) minimal ist

A*-Verfahren (II)

Ziel:

Finde einen Weg von Feld 2 nach Feld 13 (nur waagerecht und senkrecht)

Wegkosten:

nach grau = 1
nach gelb = 4

Heuristik:

h(n) = Manhattan-Distanz von n nach 13 (z.B. h(11) = 2)

- g(n) = tatsächliche Wegkosten des zurückgelegten Weges

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

Inhalt

- Motivation
- Planungsverfahren
- Planung in der Robotik
- STRIPS

Planung in der Robotik

Um zu einer geeigneten Repräsentation des Hintergrundwissens zu gelangen müssen verschiedene Bedingungen berücksichtigt werden:

- Rand- und Vorbedingungen
- Nebenbedingungen

➔ Notwendigkeit der Modellierung dieser Bedingungen

Planung in der Robotik

Vorbedingungen

Nicht alle Aktionen in bestimmten Situationen möglich

Beispiel: „Greifen“

Roboter kann Objekt greifen, wenn

- Objekt zugänglich ist
- Masse des Objekts vom Roboter bewegt werden kann
- Greifer geöffnet ist
- Greifer sich am Greifpunkt des Objekts befindet
- Greifer zum Greifen des Objekts geeignet ist

Planung in der Robotik

Nebenbedingungen

Bestimmte Restriktionen müssen eingehalten werden

Beispiel: „Montageplanung“

- min. Bearbeitungszeit oder in fester Zeitspanne
- möglichst wenig Werkzeugwechsel
- mechanische Stabilität der teilmontierten Komponenten
- Verbindungsform (z.B. Einhalten der Abbindzeiten beim Kleben)
- Materialeigenschaften (z.B. verformbar)
- Beobachtbarkeit (z.B. Kamera)
- Genauigkeit bei der gegenseitigen Positionierung von Teilen

Inhalt

- Motivation
- Planungsverfahren
- Planung in der Robotik
- STRIPS

STRIPS

STRIPS:

Stanford Research Institut Problem Solver

Entwickelt 1971 von Fikes und Nilsson

Sprache aus 2 Komponenten:

- Sprache zur Beschreibung der Welt (Zustandssprache)
- Sprache zur Beschreibung, wie sich die Welt ändert (Operatorsprache)

STRIPS - Zustandssprache

- **Relationale und konstante Symbole**

  Beispiel: auf(a, b)

  auf(,): Relation mit Wertigkeit 2

  a, b: Konstanten

- Konstanten heißen auch **Terme**
- **Atom**:

  $p(t_1, ..., t_i)$, mit p Relation, $t_1,...,t_i$ Konstanten

- **Formeln**:

  Konjunktion von Atomen

- Konstanten beschreiben Objekte der Welt
- Relationen beschreiben Zustände der Welt

STRIPS - Operatorsprache

- Zustände werden über **Operatoren o** oder über aus Operatoren bestehende **Pläne p** verändert

- Ein Operator ist ein Quadrupel $o = (u, cond, add, sub)$
  - *Bezeichnung u*: semantische Bedeutung des Operators für die Zustandsänderung
  - *Vorbedingung cond*: logische Formel, ob für aktuellen Zustand Operator anwendbar ist
  - *Eigenschaftsadditionsmenge add*: zusätzliche Formeln des neuen Zustands gegenüber ursprünglicher Zustands
  - *Eigenschaftssubtraktionsmenge sub*: fehlende Formeln im neuen Zustand gegenüber ursprünglicher Zustand

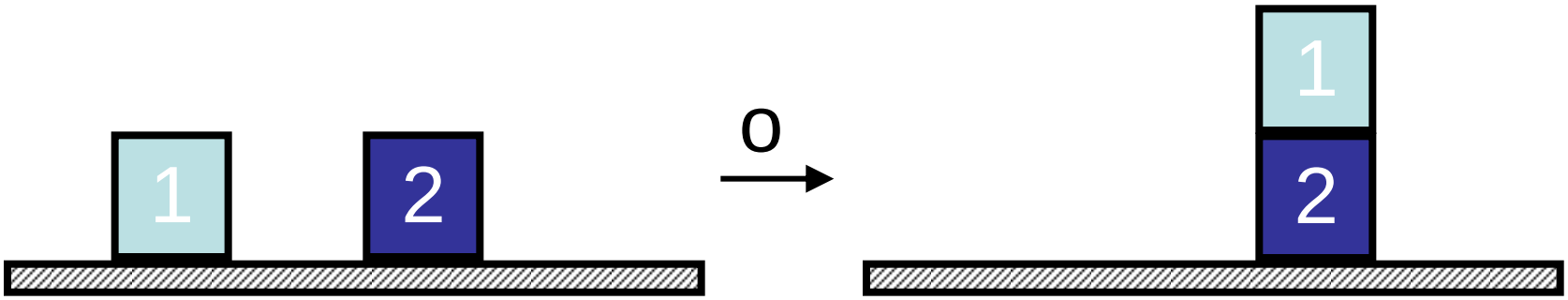
Beispiel: Anwendung eines Operators (I)

- Aktueller Zustand:

  x(y) = {frei(1), frei(2)}

- Operator-Quadrupel:

  o = *(lege_auf(1,2), {frei(1),frei(2)}, {frei(1),auf(1,2)}, {frei(2)})*

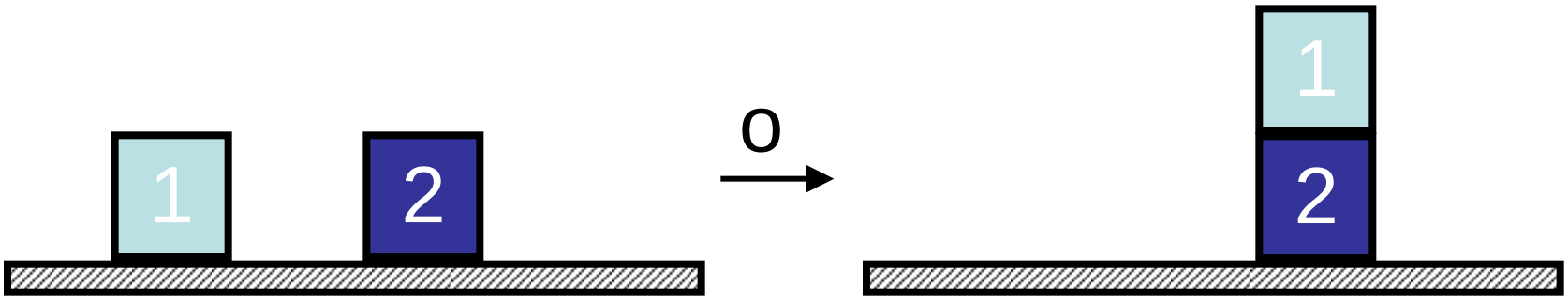


Beispiel: Anwendung eines Operators (II)

o = *(lege_auf(1,2), {frei(1),frei(2)}, {frei(1),auf(1,2)}, {frei(2)})*

- *lege_auf(1,2)* darf angewendet werden, da *{frei(1),frei(2)}* zutrifft
- Es gilt weiterhin *frei*(1), zusätzlich gilt *auf(1,2)*
- *frei(2)* gilt nicht mehr

Das Ergebnis der Anwendung des Operators ist: *x(y) = {frei(1), auf(1,2)}*



Eigenschaften der STRIPS - Operatoren

- In der *Additionsliste* müssen alle Aussageformeln stehen, die im neuen Zustand als geltend feststellbar sein müssen.
  Würde *frei(1)* fehlen, könnte lege_auf*(1,2)* in einen Zustand führen, in dem *frei(1)* nicht gilt.

- In der *Subtraktionsliste* (auch *Delete-Liste*) müssen alle Aussageformeln stehen, die im neuen Zustand nicht mehr als geltend feststellbar sein dürfen.

Problemdefinition mit STRIPS

Problem P in STRIPS ist gegeben durch Tupel:

$P = <L_S, O_S, I_S, G_S>$

mit

- L_S : Zustandssprache
- O_S : definierte Operatoren (Operatorsprache)
- I_S : Menge von Atomen, die den initialen Zustand beschreiben
- G_S : Menge von Atomen, die die Zielzustände beschreiben

Zustandsraummodell (I)

Zustandsraummodell eines Problems P mit STRIPS:

$S(P)=<S, s_0, S_G, A, \textit{next}>$

mit

S	: endliche Menge von Zuständen
$s_0 \in S$	: Startzustand
$S_G \subseteq S$	: nichtleere Menge von Zielzuständen
$A(s)$	: Menge von Aktionen, die auf den Zustand *s* anwendbar sind
next	: Übergangsfunktion, die den Zustand *s* in den Nachfolgezustand s_a = next(a, s) überführt für jede Aktion $a \in A(s)$

Zustandsraummodell (II)

- Lösung des Modells ist eine Sequenz von Aktionen a_0, a_1, ..., a_m, die den Startzustand s_0 in einen Zielzustand $s \in S_G$ überführen

- Die Sequenz generiert eine Folge von Zuständen s_i, i = 0,...,n+1 mit $s_{i+1} = next(a_i, s_i)$, $a_i \in A(s_i)$ und $s_{n+1} \in S_G$

Planen mit STRIPS

Abbildung des Planungsproblems P auf ein Zustandsraummodell

$P = <L_S, O_S, I_S, G_{S>} \Rightarrow S(P) = <S, s_0, S_G, A, \mathit{next}>$

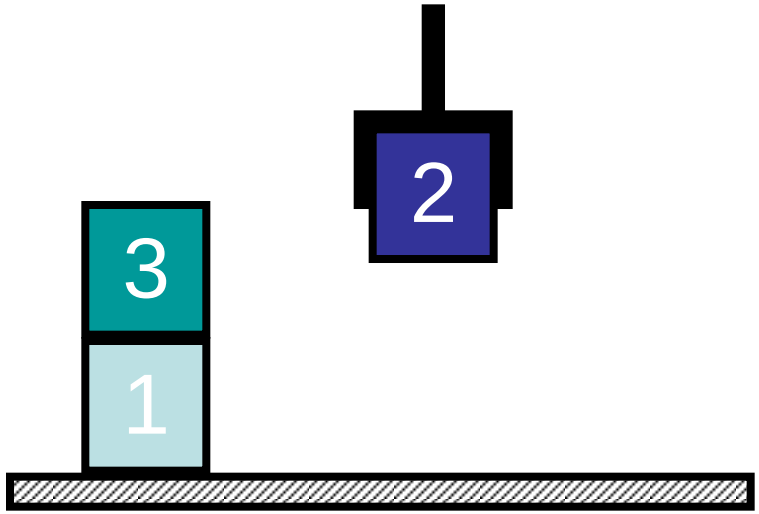
mit

- Zustände s sind Mengen von Atomen aus L_S
- Der Startzustand s_0 ist I_S
- Die Zielzustände sind alle Zustände s mit $G_S \subseteq s$
- A(s) ist die Teilmenge von Operatoren $op \in O_S$, mit $Prec(op) \subseteq s$
- Für die Überführungsfunktion *next* gilt:
  $next(a, s) = s + Add(a) - Del(a)$, für $a \in A(s)$

Beispiel (I)

Operator o = *(lege_auf(X,Y), {frei(X) ∧ frei(Y)}, {frei(X) ∧ auf(X,Y)}, {frei(Y)})*

- Handlungschema
  lege_auf(X,Y)
- Vorbedingungen
  frei(X) ∧ frei(Y)
- Effekt
  - Add = { frei(X) ∧ auf(X,Y) }
  - Delete = { frei(Y) }



Beispiel (II)

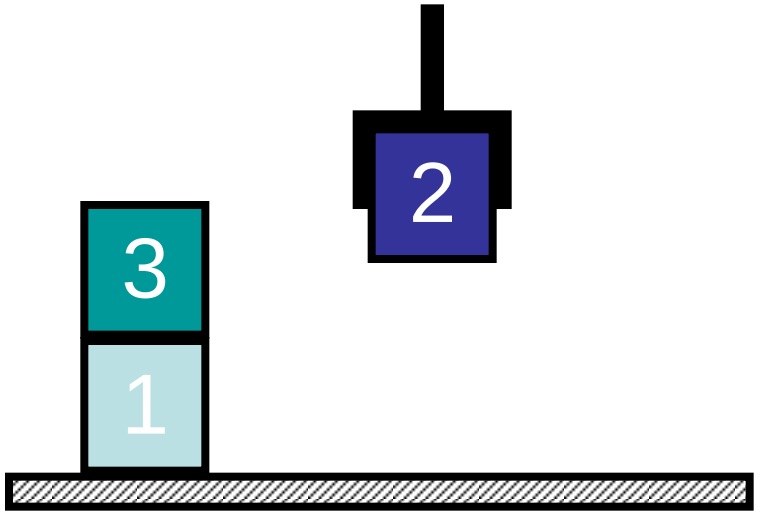
Aktueller Zustand:

auf(Block1,Tisch) $\wedge$
auf(Block3,Block1) $\wedge$
frei(Block3) $\wedge$ *frei(Block2)*

Handlungsanweisung:

lege_auf(Block2,Block3)

Zielzustand ?



Beispiel (III)

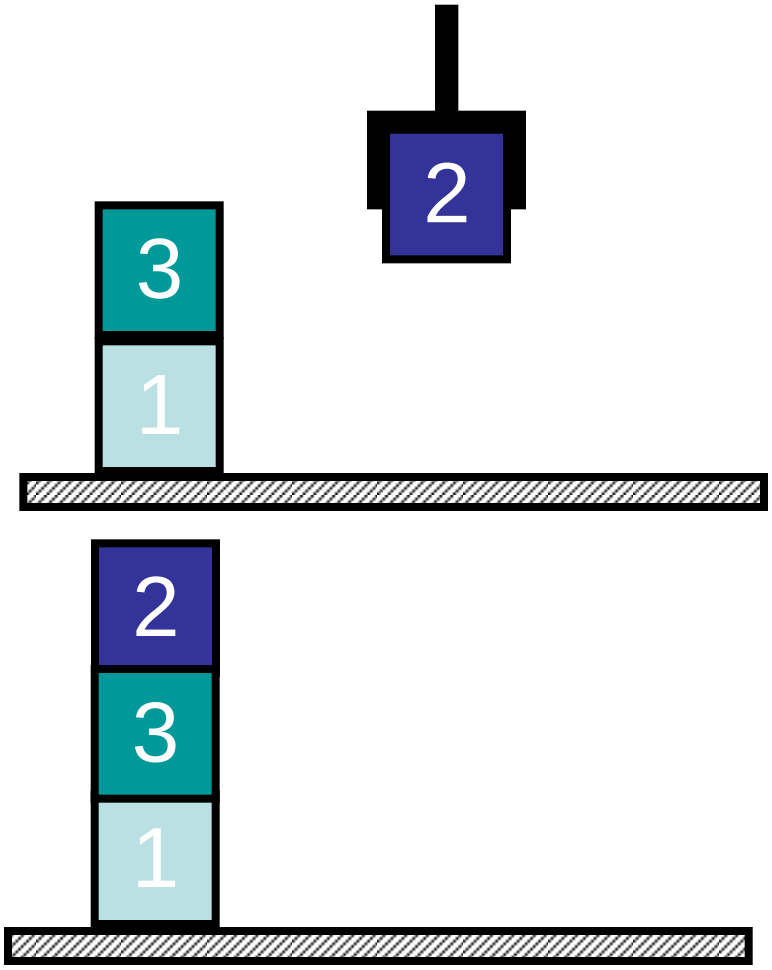
Zielzustand:

auf(Block1,Tisch) $\wedge$

auf(Block3,Block1) $\wedge$

auf(Block2,Block3) $\wedge$

frei(Block2)



Literatur

- Russel, Norvig, „Artificial Intelligence“, Kapitel 3,4
- Fikes, Nilsson, „STRIPS: A new approach to the application of theorem proving to problem solving“, In *Artifical Intelligence*. Vol. 1, Seiten 27-120, 1971
- Geffner, „Functional STRIPS: A more flexible language for planning and problem solving“

XII. Planungssysteme

Ende