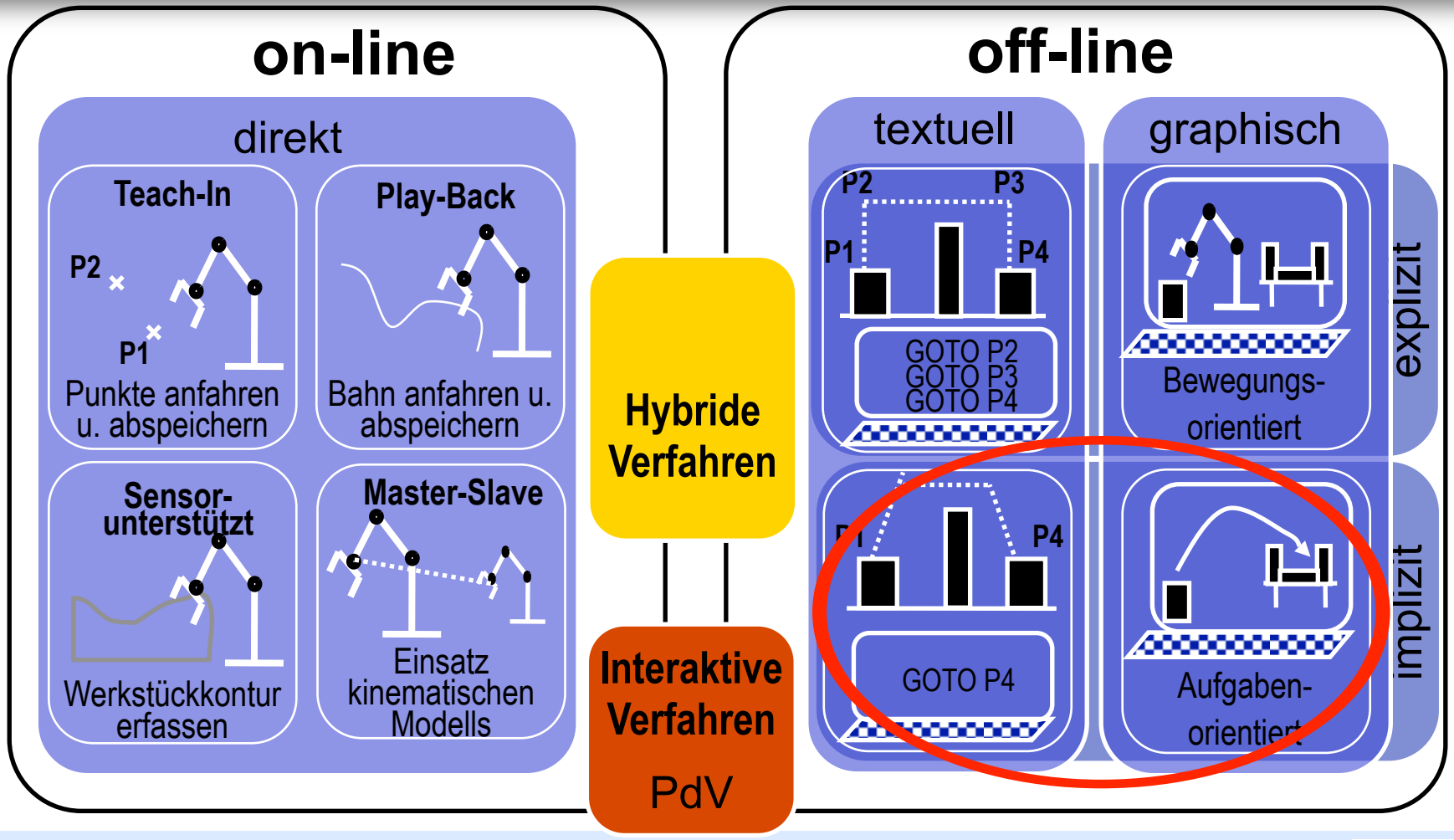
Planungsverfahren

Roboterprogrammierverfahren



Übersicht

- Definition: Planungsproblem
- Uninformierte Suche:
  - Breitensuche
  - Tiefensuche
  - Bidirektionale Suche
- Informierte Suche: Heuristiken
  - A*
  - Lineare Planung: STRIPS
  - Nichtlineare Planung: NLP-POP
  - Hierarchische Planung: HTN

Planungssystem für Roboter: Definition

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 generiert, die das betrachtete System von seinem Anfangszustand schrittweise in den gewünschten Zielzustand überführen.

The task of coming up with a sequence of actions that will achieve a goal is called planning. (Russell & Norvig: Artificial Intelligence – A Modern Approach, 2nd Edition, S. 375)

Aktionsplanung: Definition

Aktionsplanung

- Gegeben: Zielzustand, Startzustand, Menge möglicher Aktionen
- Gesucht: eine Sequenz/eine Menge von Aktionen, die den Startzustand in den Zielzustand transformieren

Alternativ:

- Erreichen einer Menge von Zielen
- Erreichen von Zielen bei gegebenen Einschränkungen
- Erreichen von Zielen mit gegebenen Richtlinien

Beispiel eines Planungsproblems

Verteilen von Post durch einen Roboter

- Gegeben: Menge an Paketen mit jeweiligem Adressaten, Menge an Adressaten, Roboteraktionen: Greifen; Fahren; Ablegen
- Gesucht: Handlungsfolge, die alle Pakete an ihren Bestimmungsort bringt

- Einschränkungen: Batterieladung
- Richtlinien: Zeitoptimal, wegoptimal, unterschiedliche Prioritäten, …

Probleme bei der Aktionsplanung

- Teilweise unbekannte Umwelt
- Messunsicherheiten
- Effekte von Aktionen nicht immer deterministisch
- Externe Ereignisse können das Ergebnis beeinflussen
- Eigene Aktionen können das Ergebnis negativ beeinflussen
- Randbedingungen (Zeit, Ressourcen, …)
- Beachtung von Richtlinien

Ansatz

Explizite Repräsentation von

- Zustand
- Zielen
- Aktionen
- Plänen

Flexible Planungsalgorithmen und Suchstrategien

- Ordnung des Planungsproblems nicht notwendigerweise Ordnung der Planausführung

Zerlegen des Ziels

- Teilziele können mehr oder weniger unabhängig erreicht werden

Vereinfachende Annahmen

- Bekannter Ausgangszustand
- Deterministische Aktionen
- Keine externen Störungen
- Einfache Aktionsbeschreibung:
  - Keine bedingten Effekte
  - Keine quantifizierten Effekte
  - Keine funktionalen Effekte
- Aktionen nur sequentiell
- Keine Aktionen auf Sensorik (→keine Verzweigungen im Plan)
- Keine Zeitbeschränkung
- Ausreichende Ressourcen

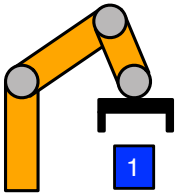
Planen als Suchproblem

- Definition eines Suchproblems
  - Anfangszustand
  - Zielzustände / Zieltest
  - Nachfolger-Funktion
  - (Zustandsraum)
  - Gütekriterium für Lösungen (optional)

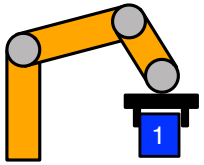
- Lösungen stellen Pfade im Zustandsraum dar
- Suchverfahren unterscheiden sich in den Expansionsstrategien

Beispiel

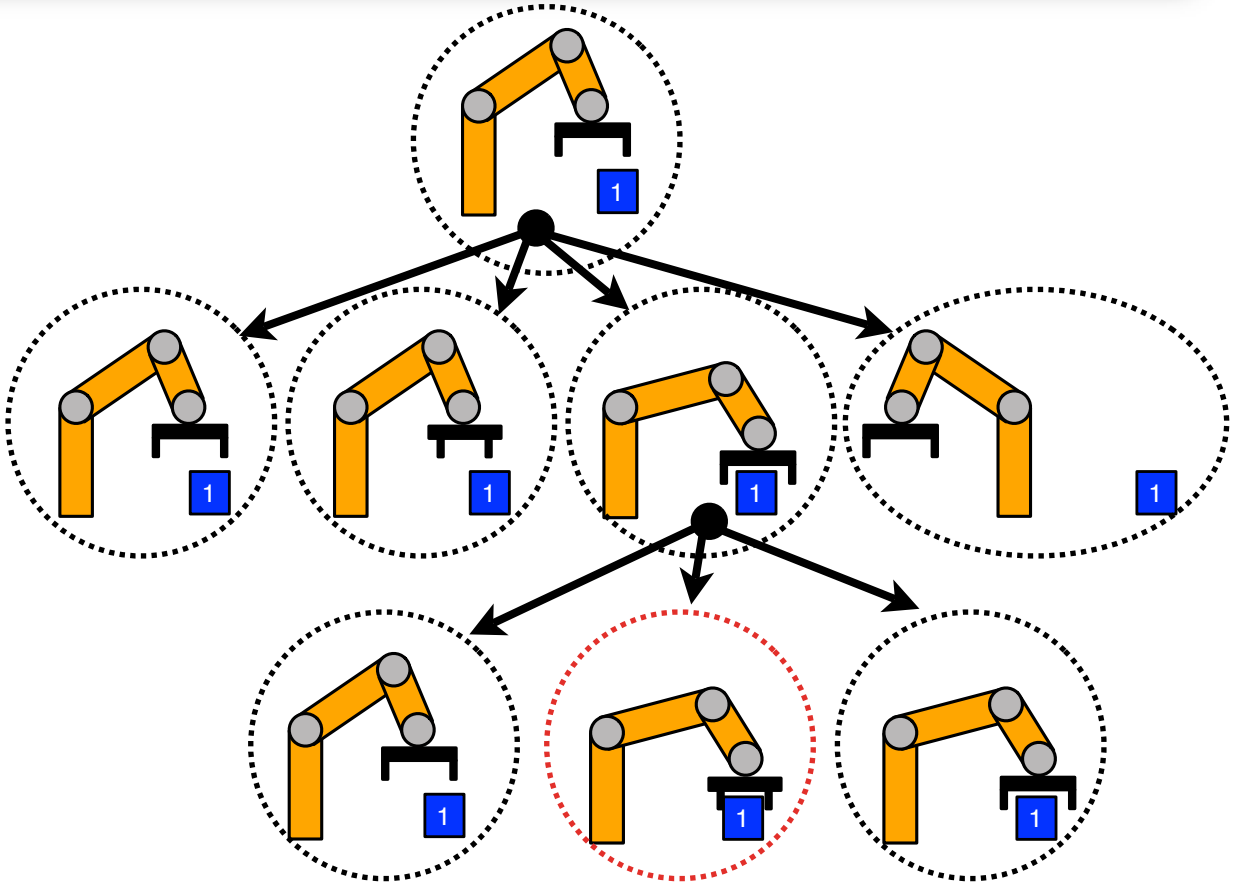
Startzustand:



Zielzustand:



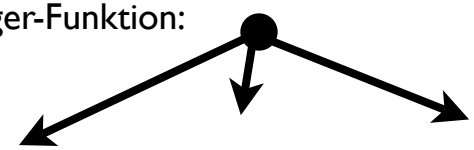
Aktionen:
1. Roboterarm links, rechts, hoch, runter
2. Roboterhand zu, auf



Zustände:

Ziele:

Nachfolger-Funktion:



Suchverfahren

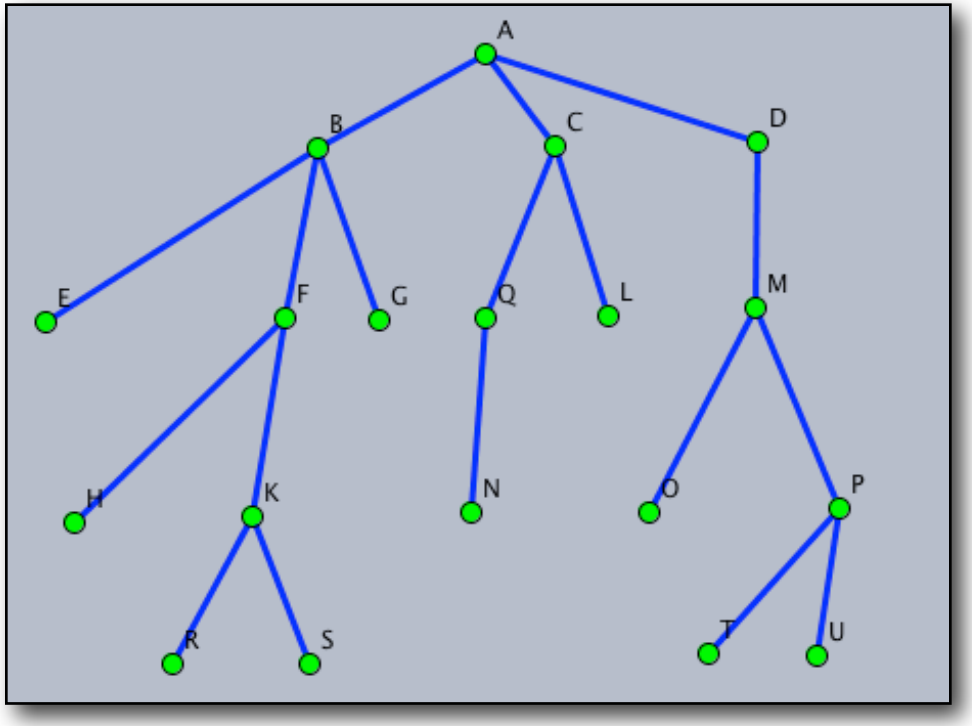
- Uninformiert: Analytische Verfahren (Baumsuche)
  - Breitensuche
  - Tiefensuche
  - Bidirektionale Suche

- Informiert: Heuristiken
  - A*
  - Dekomposition (Zerlegung in unabhängige Teilprobleme)
  - Aufgabenspezifisches Wissen erforderlich

Bewertung von Suchverfahren

- Vollständigkeit
  - Wenn eine Lösung existiert, wird sie auch gefunden
- Optimalität
  - Wenn eine Lösung gefunden wird, ist diese auch optimal bezüglich eines Gütekriteriums
- Zeit Komplexität
  - Wie lange dauert es eine Lösung zu finden
- Raum Komplexität
  - Wieviel Speicher wird für die Suche benötigt

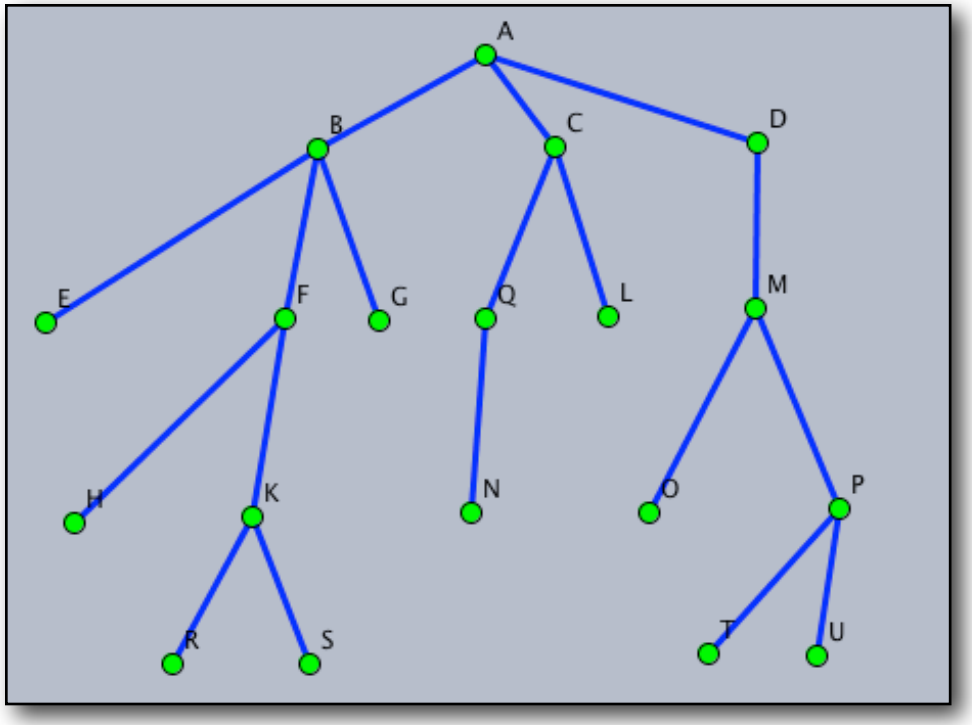
Suchverfahren: Breitensuche



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

- Vollständig

Suchverfahren: Tiefensuche

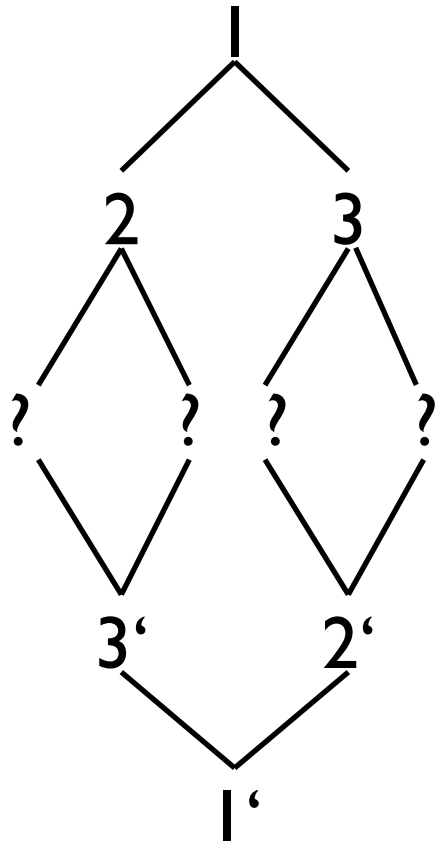


- Beginn am Ausgangszustand
- Suche bis zur maximalen Tiefe
- Weiter im tiefsten Knoten, der noch nicht durchsuchte Kanten hat

- Nicht vollständig*

* falls Suchraum und Suchtiefe unbeschränkt sind

Suchverfahren: Bidirektionale Suche



- Beginne eine Breitensuche am Ausgangszustand und eine weitere am gewünschten Zielzustand
- Halte, wenn bei beiden Suchen ein gleicher Zustand erreicht wurde
- Die Lösung setzt sich zusammen aus den beiden Pfaden, mit denen dieser gleiche Zustand erreicht wurde
- Vollständig

Suchverfahren: Progression ↔ Regression

Progression:

- Wähle Aktion, deren Vorbedingungen erfüllt sind
- Fortfahren, bis Zielzustand erreicht ist

+ Einfacher Algorithmus
- Suche kann sehr breit werden

Regression:

- Wähle Aktion, deren Effekt ein nicht erfülltes Teilziel erfüllt
- Füge nicht erfüllte Vorbedingungen der Aktion zu den Teilzielen hinzu
- Fortfahren, bis keine unerfüllten Teilziele mehr existieren

+ Fokus auf der Erfüllung von Teilzielen
- Regression ist unvollständig für funktionale Effekte

Informierte Suchverfahren: Heuristiken

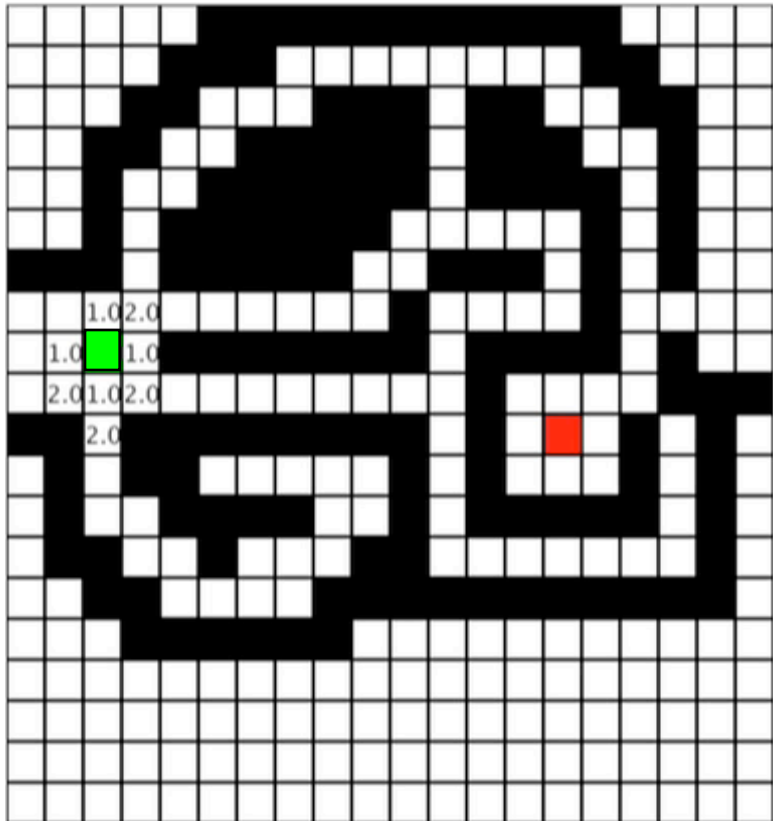
Möglich, wenn zusätzliches Wissen zugänglich ist, z.B.

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

Suchverfahren: A*

- 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
- vollständig
- optimal für den Fall, dass h(n) ein untere Schranke für die tatsächlichen Kosten darstellt

Beispiel: A*



Start

Ziel

h(n) = Manhattan-Distanz

http://www.vision.ee.ethz.ch/~cvcourse/astar/AStar.html

Dekomposition in Teilziele: Lineare Planung ↔ Nichtlineare Planung

Linear:

- Lösungen von Teilzielen werden nacheinander geplant
- Stack (Stapel) noch offener Teilziele

+ Einfache Suchverfahren
+ Effizient, wenn Teilziele unabhängig
- Kann suboptimale Pläne erzeugen
- Unvollständig

Nichtlinear:

- Verschachteltes Lösen der Teilziele
- Menge noch offener Teilziele

+ Vollständig
+ Kürzere (bessere) Pläne möglich
- Größerer Suchraum

Lineare Planungsmethoden

Ansatz:

- Jedes Teilziel wird für sich gelöst. Dann Lösung des nächsten Teilziels
- Teilziele auf einem Stack
- Vorraussetzung: Teilziele sind (weitestgehend) unabhängig
- Lineare Planung ist effizient, wenn diese Vorraussetzung erfüllt ist

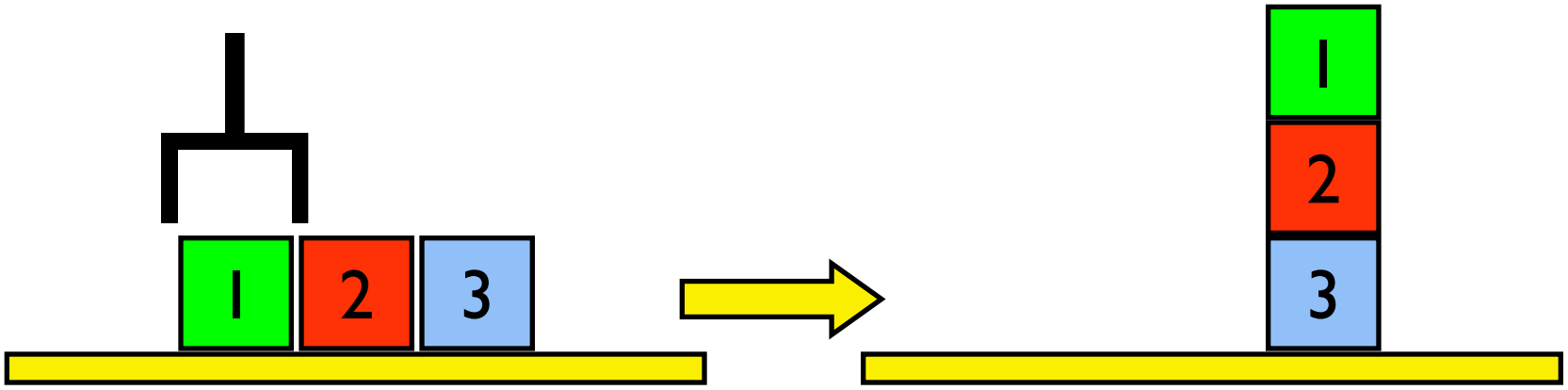
Klassisches Beispiel: Die Blockwelt

Standardbeispiel: Einfach und umfassend

Szenario: Tisch mit Blöcken, Greifer

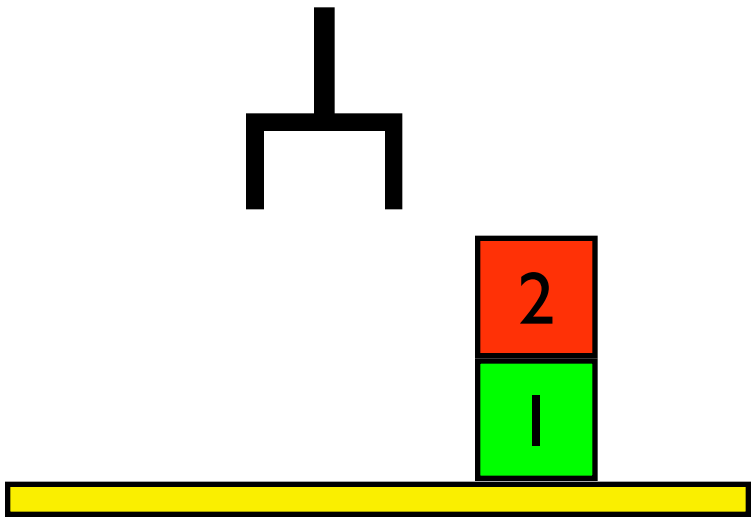
Operatoren:

- Pickup(x) Block vom Tisch greifen
- Putdown(x) Block auf Tisch ablegen
- Stack(x,y) x auf y ablegen
- Unstack(x,y) x von y aufnehmen



Formalisierung: Situationskalkül

- Szenenmodell = Konjunktion von Prädikaten in Prädikatenlogik
- Beispiel einer Situation zum Zeitpunkt S0
  - Ontable(1, S0) $\wedge$ On(2, 1, S0) $\wedge$ Holding(NIL, S0) $\wedge$ Clear(2, S0)
- Gesucht: Situation mit
  - $\exists$t: [Holding(2, t)]
- ($\rightarrow$2 mit Greifer nehmen)



Formalisierung der Aktionen

Aktion:

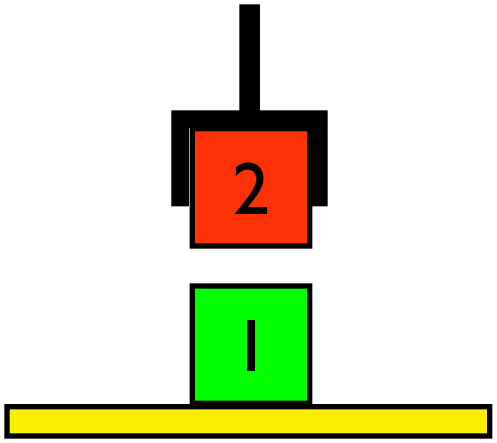
- Stack(x,y, St)

Vorbedingungen:

- Holding(x, St) $\wedge$ Clear(y, St)

Nachbedingungen:

- $\neg$Holding(x, St+1) $\wedge$ Holding(nil, St+1) $\wedge$
  $\neg$Clear(y, St+1) $\wedge$ On(x,y, St+1) $\wedge$ Clear(x, St+1)



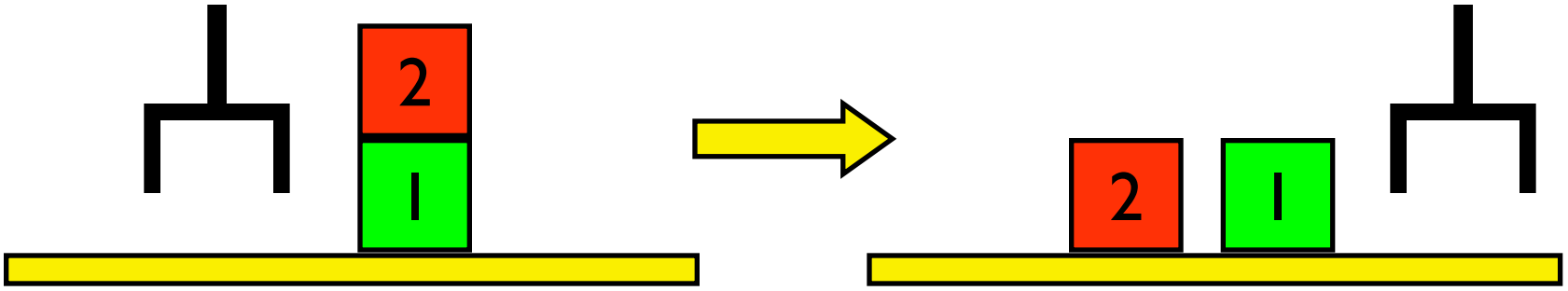
Beispiel: Lineare Planung in der Blockwelt

Initial (0):
Ontable(1, S0) ∧ On(2, 1, S0) ∧ Holding(NIL, S0) ∧ Clear(2, S0)

Ziel (t):
Ontable(1, St) ∧ Ontable(2, St) ∧ Holding(NIL, St) ∧ Clear(1, St) ∧ Clear(2, St)

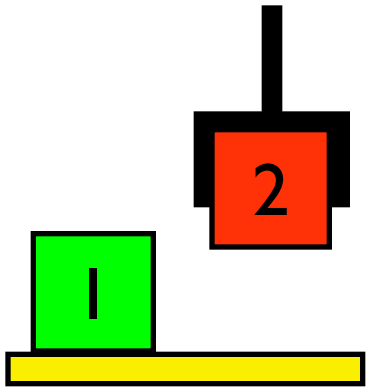
Planer:

- 1. Ziel (Ontable(2,St)): Unstack(2, 1, S0); Putdown(2, S1);
- 2. Ziel (Clear(1, St)): Wird per Zufall mit 1. Ziel abgedeckt ✓



Formalisierung der Blockwelt-Operatoren: Rahmenproblem (Frame Problem)

- Operator Putdown(x, St):
  - Vorbedingung: Holding(x, St)
  - Nachbedingung: Ontable(x, St+1) ∧ Clear (x, St+1) ∧ Holding(NIL, St+1) ∧ ¬Holding(x, St+1)



- Beispiel: Zustand S0:
  - Holding(2, S0) ∧ Ontable(1, S0)

- Nach Anwendung von Putdown(2, S0) gilt Ontable(2, S1)
- Intuitiv ist klar: Es gilt auch Ontable(1, S1). Dies kann aber nicht formal aus Ontable(1, S0) abgeleitet werden.
- Also: Ontable(2, S1) und Ontable(1, S1) kann nicht abgeleitet werden

Rahmenaxiome

- Bisher: Notation, was ein Operator bewirkt
- Zusätzlich: Was ein Operator nicht verändert
- Also: Nachbedingungen der Operatoren müssen alle möglichen Nicht-Veränderungen (Rahmenaxiome) enthalten
- Beispiel:
  Für alle x,y,s [Ontable(x,s) → Ontable(x, Unstack(y,x,s))]

  → Lästig! Modellierung und Planen darauf zu komplex

STRIPS

- STRIPS:
  STanford Research Institute Problem Solver (Fikes & Nilsson 1971)
- Linearer Planer

- Wie bisher:
  - Operatoren haben Vor- und Nachbedingungen
- Nachbedingungen bestehen aus Add- und Delete-Liste
  - Add-Liste: Merkmale, die der Operator erzeugt
  - Delete-Liste: Merkmale, die der Operator zerstört

STRIPS-Annahme

Ein Operator ändert, wenn man ihn in der Welt ausführt, genau das, was in seiner Nachbedingung angegeben ist.

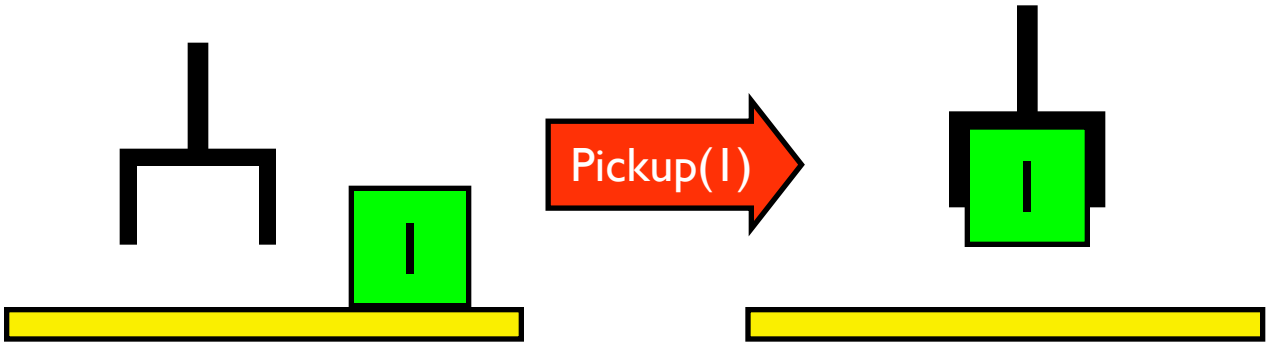
STRIPS: Operator

Operator: o
Aktion: Pickup(x)
Vorbedingungen:

- Holding(NIL) $\wedge$ Clear(x) $\wedge$ Ontable (x)

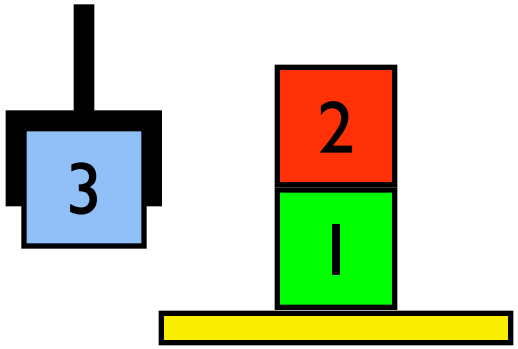
Nachbedingungen:

- Add-Liste add(o):
  - Holding(x)
- Delete-Liste del(o):
  - Holding(NIL)
  - Clear(x)
  - Ontable(x)



STRIPS: Planung

- Situation: Ansammlung von Fakten:
  { Ontable(1), On(2, 1), Holding(3), ... }
- Planung: Suche eines Weges vom Initialzustand zum Zielzustand
- Ziel: Nicht ein Zielknoten, sondern Menge von Knoten
  → Ziel gibt nicht immer alles vor
  (Ziel: On(3, 2) sagt nichts aus über 1)



- Günstig:
  Rückwärtssuche von der Menge der Zielknoten → zielzentriert

STRIPS: Planung

- Planen als Rückwärtssuche mit Zielkeller (MZAMZK, Mittel-Ziel Analyse mit Zielkeller)

- Eingabe: Z: Menge der Ziele
  S: Startsituation
- Global: Menge der Operatoren
- Variable: St: Aktuelle Situation
- Ausgabe: P: Plan (lineare Ordnung von Operatoren)

STRIPS Algorithmus (MZAMZK)

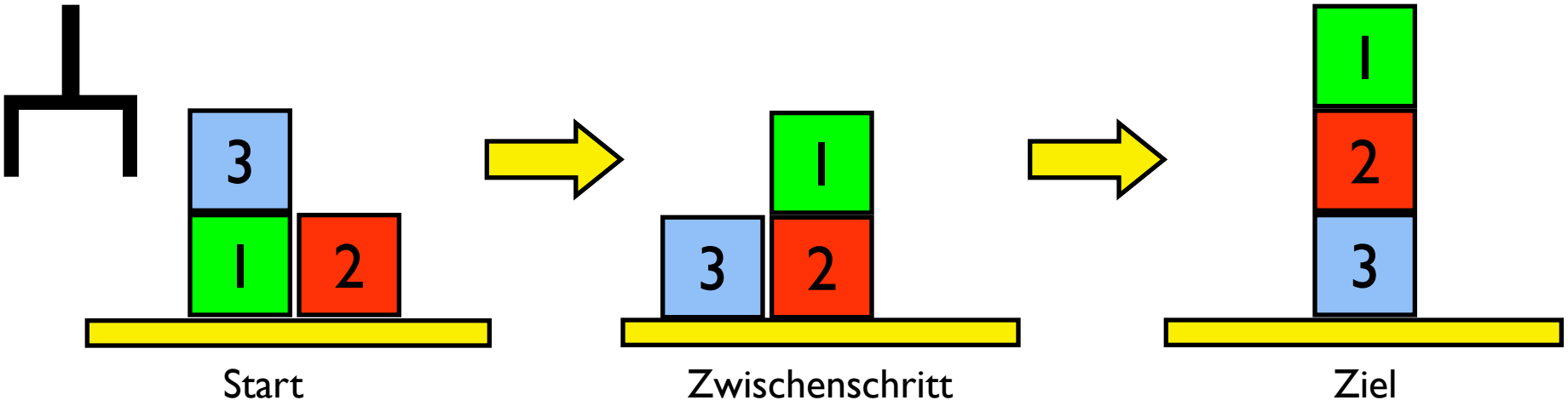
Wiederhole solange offene Ziele vorhanden

- Wähle das erste Ziel g aus (top of stack)
- Wähle einen anwendbaren Operator o, dessen Add-Liste g enthält
- Falls kein solcher Operator existiert dann Backtracking und weiter bei 1.
- Andernfalls füge alle nicht erfüllten Vorbedingungen von o zu den Zielen hinzu (push on stack)

Lineare Planungsverfahren: Suboptimale Pläne

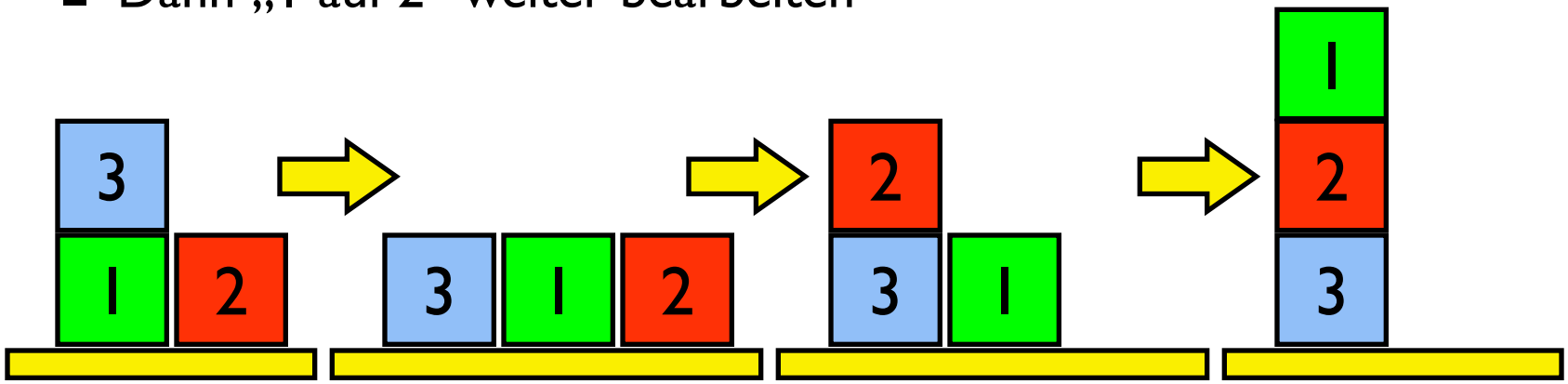
Sussmann Anomalie

- Ziel: On(1,2) ∧ On(2,3)
- Erzeugter Plan (mehrere Lösungen):
  Unstack(3,1); Putdown(3); Pickup(1); Stack(1,2); Unstack(1,2); Putdown(1); Pickup(2); Stack(2,3); Pickup(1); Stack(1,2)



Lineare Planungsverfahren: Teilzielinteraktion

- Zur Lösung müsste man
  - Erst „1 auf 2“ bearbeiten
  - Dann „2 auf 3“ bearbeiten
  - Dann „1 auf 2“ weiter bearbeiten



- Verzahnung von Teilzielen bei der linearen Planung nicht möglich

Lineare Planungsverfahren: Teilzielinteraktion

Unterscheidung in negative und positive Teilzielinteraktion

- Negative TZI: Bearbeitung eines Teilziels zerstört Teile der Bearbeitung eines anderen Ziels
- Positive TZI: Bearbeitung eines Teilziels erledigt Teile der Bearbeitung eines anderen Ziels mit

STRIPS berücksichtigt positive TZI, aber keine negative TZI

Erweiterung von MZAMZK: Kritiker

- Einführung eines Kritikers zum Plan-Debugging
- Finden von sich aufhebenden Operationen
- Löschen der entsprechenden Planteile

- Sussmann Anomalie:
  Pickup(1); Stack(1,2); Unstack(1,2); Putdown(1);
- Stack(1,2); Unstack(1,2); hebt sich auf -> löschen
- Jetzt: Pickup(1); Putdown(1); direkt hintereinander! → löschen
- Optimierter Plan:
  Unstack(3,1); Putdown(3); Pickup(2); Stack(2,3); Pickup(1); Stack (1,2)

Unlösbare Probleme: One way Rocket- und Zuweisungs-Problem

One way Rocket-Problem (hier: Postzustellung):

- Aufnehmen(roboter, paket, ort)
- Abgeben(roboter, paket, ort)
- BringenLaborgebäude(roboter, von, nach)
  - Pre: …, Batterie-Voll(roboter)
  - Add: … Delete: Batterie-Voll(roboter)

Initial: An(brief1, hauptgebäude) $\wedge$ An(brief2, hauptgebäude) $\wedge$ Batterie-Voll(albertII)

Ziel: An(brief1, laborgebäude) $\wedge$ An(brief2, laborgebäude)

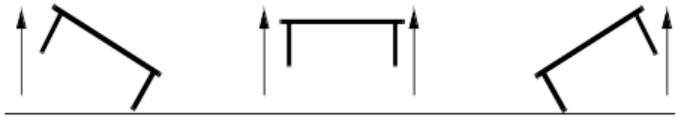
Nicht lösbar mit linearer Planung!

Lineare Planungsverfahren: Diskussion

Vorteile:

- Reduzierter Suchraum, weil Teilziele nacheinander gelöst werden
- Vorteile, wenn Teilziele unabhängig
- Produziert nur gültige Pläne

Nachteile:



- Kann suboptimale Pläne erzeugen
- Lineare Planung ist unvollständig
  (Es wird nicht immer eine Lösung gefunden, wenn eine existiert)

Nichtlineare Planungssysteme

- Idee: Ziel-Menge statt Ziel-Stack
- Im Suchraum werden alle möglichen Teilziel-Anordnungen dargestellt
  - Teilzielinteraktion durch Ziel-Verzahnung bei der Planung betrachtet
- Vorteile (mit entsprechenden Planungsverfahren):
  - Erzeugt nur gültige Pläne
  - Vollständig
  - Optimalität bzgl. irgendeines Kriteriums kann erreicht werden
- Nachteile:
  - Größerer Suchraum, alle Teilzielreihenfolgen müssen beachtet werden
  - Komplexere Algorithmen

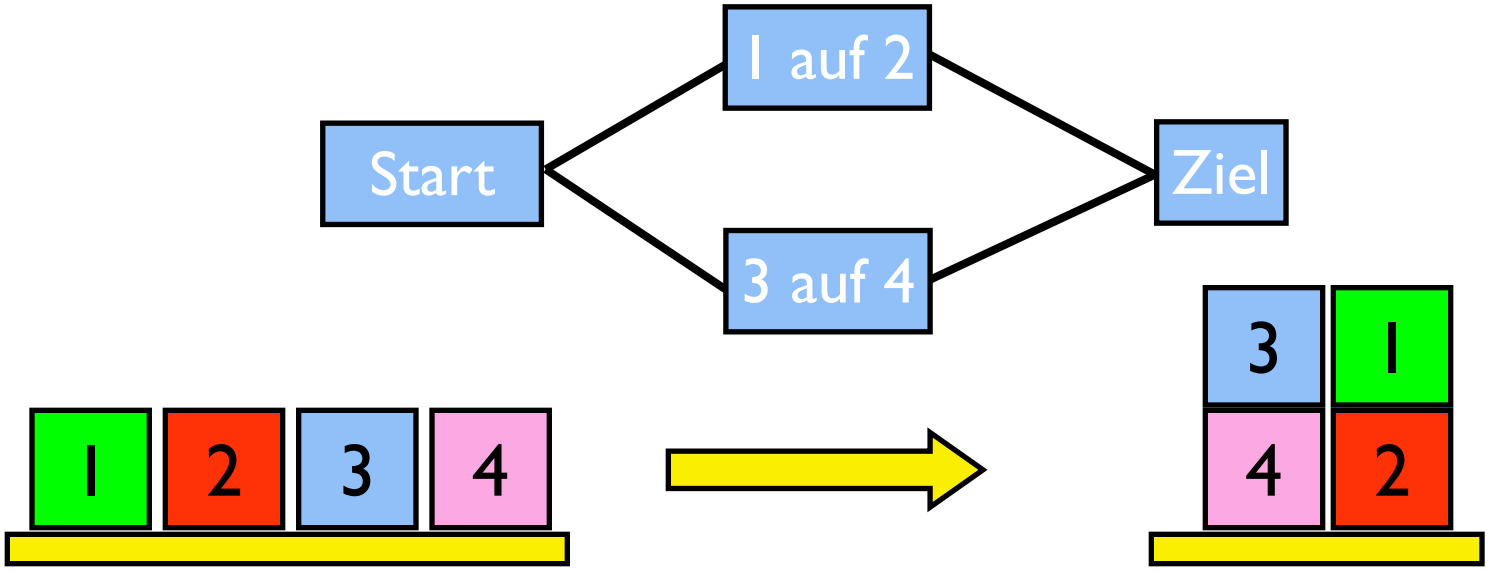
Nichtlineare Planungssysteme: Ein einfacher Algorithmus

Wiederhole, solange Zielmenge G nicht leer

- Wähle g aus G
- Wenn g nicht im Zustand enthalten
  - Wähle Operator o, dessen Add-Liste g enthält
  - Füge o zu den auszuführenden Operationen hinzu
  - Füge Vorbedingungen von o zu G hinzu

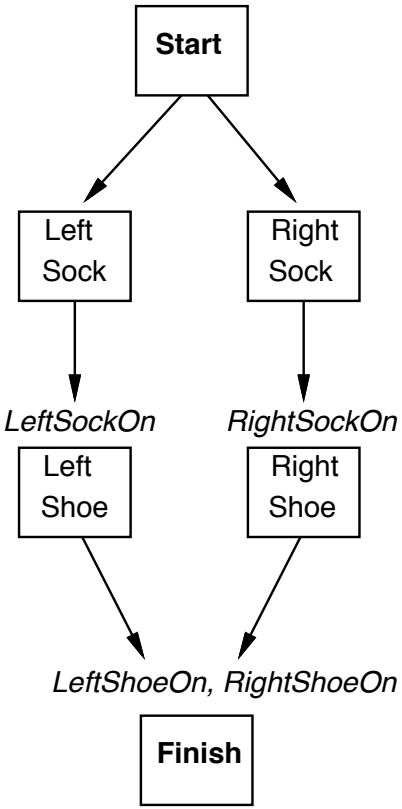
Nichtlineare Planungssysteme

- Abhängigkeiten der einzelnen Operatoren können geplant werden
- Plan kann z.B. in Form eines Vorranggraphen gespeichert werden
  → Partial Order Planning (POP)

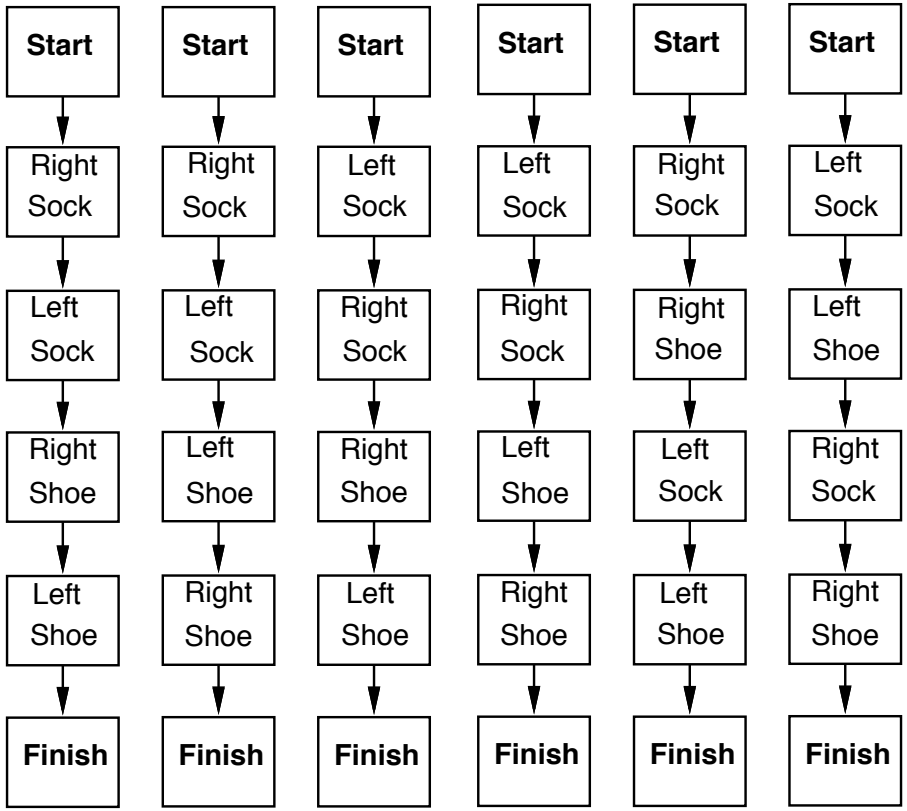


Planen von Teilordnungen: Beispiel

Partial Order Plan:

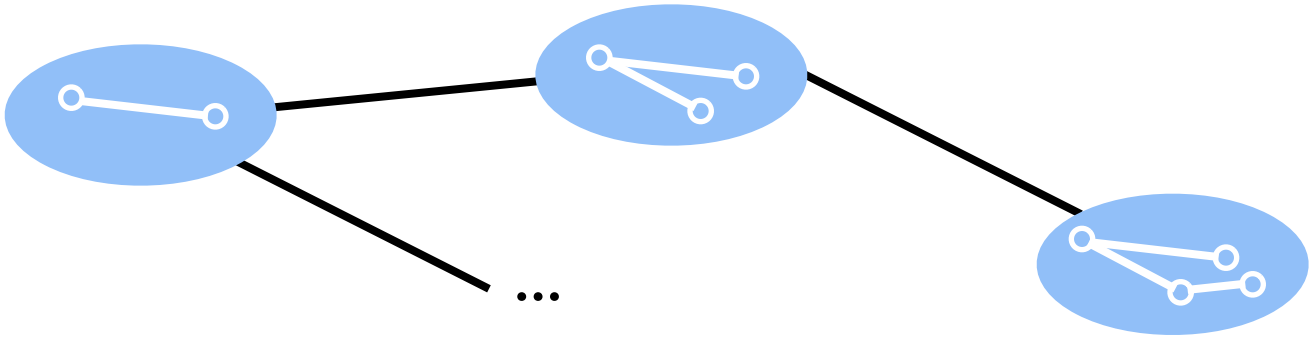


Total Order Plans:



NLP POP (1)

Partial Order Planning (POP) - Suche im Planraum



Knoten: (Unvollständige) Pläne

Kanten: Planmodifikationsschritte

Planmodifikationsschritte:

- Einfügen von Planschritten
- Anordnen von Planschritten
- Instanziieren von Variablen

NLP POP (2)

NLP folgt Least Commitment Strategie: „Prinzip der geringsten Festlegung“

- Es gibt keine a priori Festlegung der Reihenfolge, in der die Ziele erreicht werden sollen
- Die Reihenfolge, in der Aktionen ausgeführt werden, wird nur soweit nötig festgelegt
- Variablen werden nur, falls notwendig, instanziiert

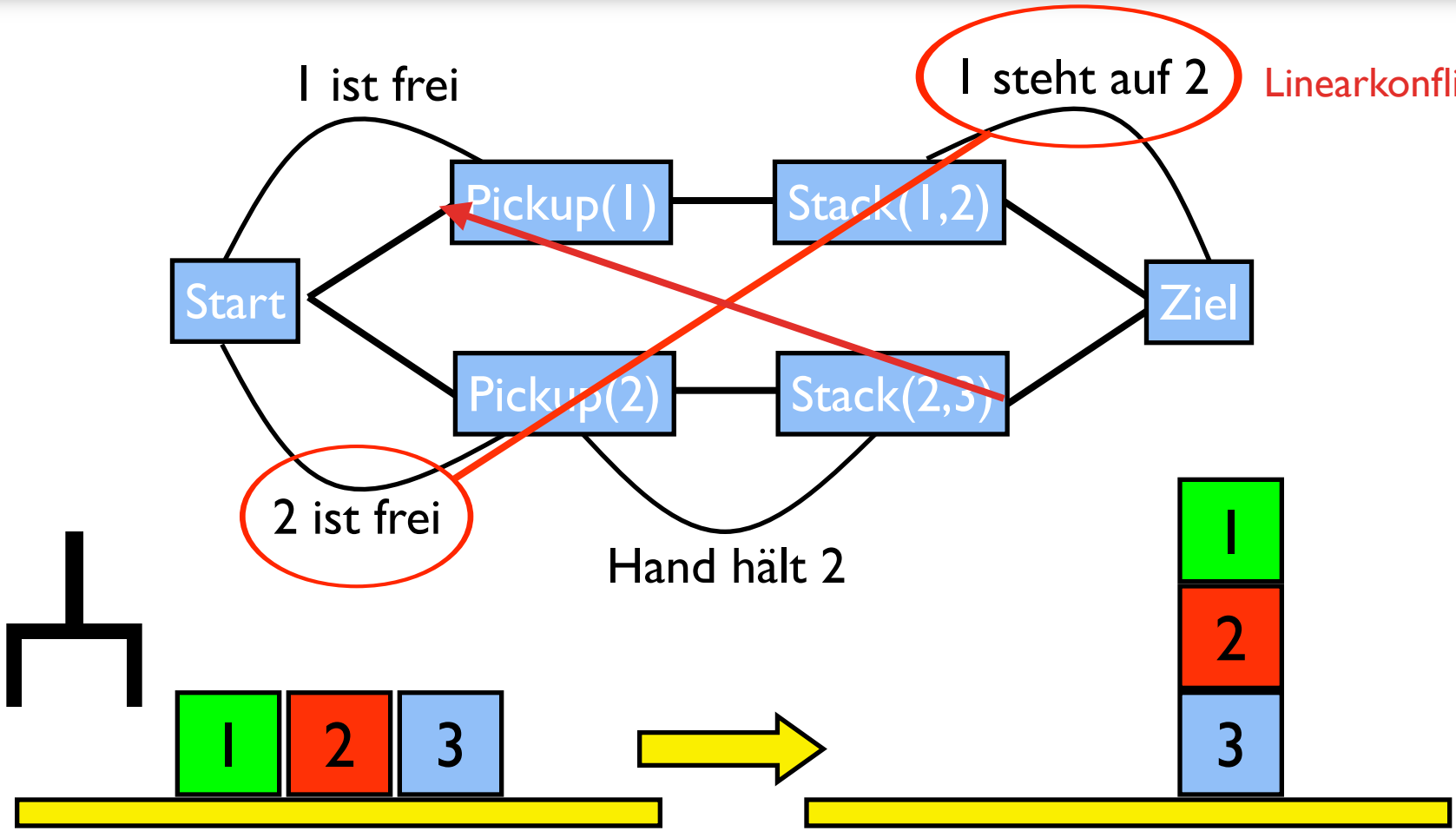
NLP POP (3)

POP ist vollständig und erzeugt konsistente nicht-lineare Pläne. Jede Linearisierung eines solchen Plans ist eine Lösung des Planungsproblems.

Ein nicht-linearer Plan P ist eine Datenstruktur mit folgenden Komponenten:

- eine Menge von Planschritten S mit Operation o (S:o)
- einer Menge von Ordnungsbedingungen, die die Reihenfolge von Planschritten angeben (S_i < S_j), d.h. „S_i vor S_j“
- einer Menge von Variablenbedingungen (x=t), wobei x eine Variable und t ein Term ist, und Ungleichungsbeschränkungen
- einer Menge von kausalen Beziehungen (causal links), die Zusammenhänge zwischen Planschritten beschreiben, d.h. S_i erzeugt die Vorbedingung c für S_j
- Menge der offenen Vorbedingungen

Nichtlineare Planungssysteme POP: Konflikte



NLP: Kritiker

Idee: Nach der Konfliktauflösung den Plan weiter optimieren.

Beispiele:

- Redundante Kanten entfernen
- Operatoren löschen, von denen keine Abhängigkeiten ausgehen
- Hilfreiche Interaktionen entdecken
- Umwege entfernen (siehe Sussmann-Anomalie bei LP)

Total Order ↔ Partial Order Planning

Total Order:

- Plan entspricht immer einer strikten Sequenz von Aktionen

+ Einfachere Algorithmen
- Keine nebenläufigen Pläne
- Enthält evtl. unnötige Abhängigkeiten

Partial Order:

- Aktionen können ungeordnet sein
- Nur notwendige Abhängigkeiten werden geplant
- Plan muss evtl. vor der Ausführung linearisiert werden

+ Enthält nur notwendige Abhängigkeiten
+ Nebenläufige Pläne darstellbar
- Feststellung der erfüllten Teilziele zu einem Zeitpunkt schwierig
- Komplexere Darstellung

Hierarchische Planungsverfahren

Hierarchische Aufgabennetzwerke (Hierarchical Task Networks, HTN)

Grundsätzlicher Ansatz:

- Komplexe Pläne haben meistens erkennbare Strukturen
- Diese Struktur kann oft in Form von Hierarchien ausgedrückt werden
- Teilpläne sind oft voneinander unabhängig
- Hierarchische Planung versucht, die Dimensionalität des Problems zu reduzieren, im Gegensatz zur vollständigen Suche im Plan- oder Zustandsraum

Beispiel

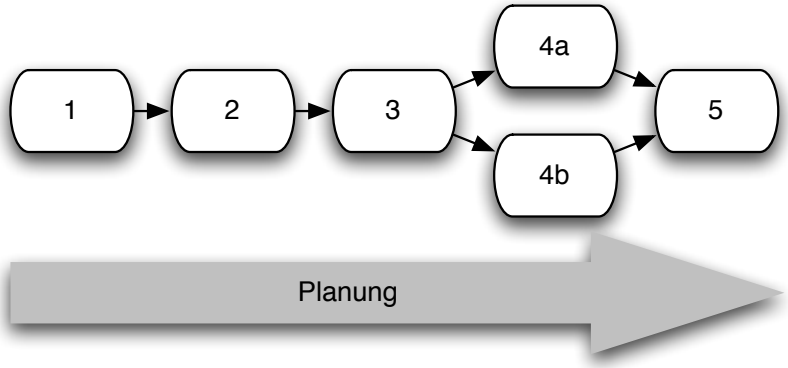
Um zur Konferenz nach X zu kommen, muss man zum Flughafen kommen, ein Flugzeug nach X nehmen, zum Konferenzhotel fahren.

Um zum Flughafen zu kommen, kann man entweder einen Bus nehmen oder ein Taxi.

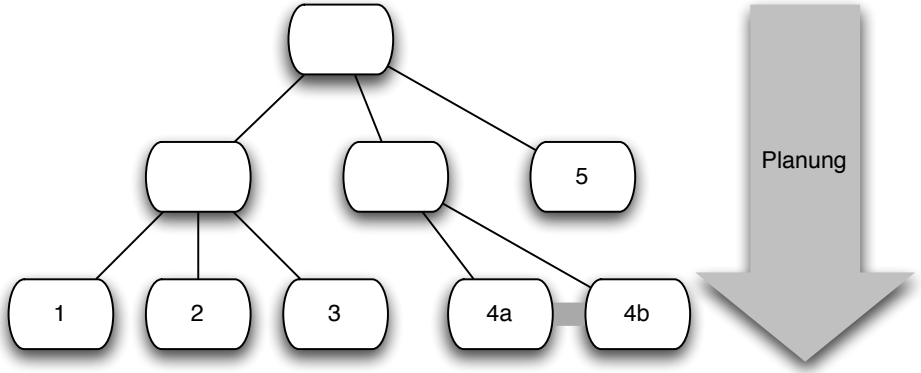
Wenn man genug Geld hat:
Um ein Taxi nach Y zu nehmen, entweder vorher anrufen, oder ein Taxi herbeiwinken, einsteigen, sage „Ich möchte nach Y“, warte bis Y erreicht ist, bezahlen, aussteigen.

Vergleich mit STRIPS-Derivaten

Planungsrichtung unterschiedlich:
Klassische Planung kombiniert elementare Operationen,
hierarchische Planung „entfaltet“ abstrakte Operatoren



Klassische Planung



Hierarchische Planung

Abstrakte Pläne

- Primitive Operatoren
  - STRIPS-ähnliche Beschreibung
  - keine Vorbedingungen
- Zusammengesetzte Operatoren
  - Vorbedingungen, Effekte
  - Methoden zur Zerlegung in Teilpläne
    - Aufbau der Teilplanstruktur
    - Parameter
    - Ähnlich Funktionsaufrufen
- Ein Abstrakter Plan enthält zusammengesetzte Operatoren
- Ein voll instantiierter Plan besteht nur aus Primitiven Operatoren

HTNs: Suchraum

- Ziel ist eine abstrakte Aufgabe (Task), nicht ein Weltzustand (State)
- Operatoren im Suchraum
  - Zerlegung in Teilpläne
  - Parametrisierung
  - Konfliktlösung
- Algorithmus HTN-Planung:
  - Starte mit initialer abstrakter Aufgabe (nicht Weltzustand/Zielliste!)
  - Baue das Aufgabennetzwerk durch wiederholtes Expandieren in Teilpläne auf, bis der Plan voll instantiiert ist
  - Expandieren durch Verwendung von Methoden, deren Anwendbarkeit gegeben ist

Beispiel: Simple Hierarchical Order Planner (SHOP) Nau et al., 1999

SHOP Algorithmus:

- Vorwärtssuche, lineares Planungsverfahren
  - Planung in Ausführungsreihenfolge
  - Tiefensuche
- Elementaroperatoren haben keine Vorbedingungen
- Keine parallelen Aktionen
- Sehr mächtige Operatorrepräsentation
- Effizienter Planungsalgorithmus
- Korrekt und vollständig

SHOP Beispieldomäne

```
(:operator (!putdown ?block)
   ((holding ?block))
   ((ontable ?block) (handempty)))
```

Delete Liste
Add Liste

```
(:method (make-clear ?y)
   ((clear ?y))
   nil)
```

Vorbedingung
Task-Liste

```
(:method (make-clear ?y)
   ((on ?x ?y))
   ((make-clear ?x) (!unstack ?x ?y) (!putdown ?x)))
```

Vorbedingung
Task-Liste

HTN: Where is the power?

- Methoden codieren Domänenwissen
- Methoden beinhalten Problemlösungswissen
- Abstraktion kapselt Operatorinteraktionen

- Vorsicht bei:
  - Immer noch NP-vollständig
  - Terminiert nicht zwangsläufig (rekursives Anwenden von Methoden, Endlosschleifen sind evtl. schwer zu detektieren!)
  - Plan kann erst im voll expandierten Zustand bewertet werden
    → Konflikte können evtl. erst dann entdeckt werden

Mögliche Erweiterungen zu HTN

Erkennung/Lösen von Konflikten

- Teilzielinteraktion
- Mehrfachverwendung/Wiederverwendung von Operatoren

Methoden beinhalten Vorbedingungen und Effekte

- Konflikte in der Planungsphase erkennen

Methoden beinhalten Ressourcenangaben

- Scheduling

Zusammenfassung

- Planung: Auswahl einer Sequenz von Aktionen (Operatoren), die einen Initialzustand in einen Zielzustand überführen
- Uninformierte und informierte Suche
- Lineare Planung: Planung als Suche mit einem Teilziel-Stack
- Nichtlineare Planung: Planung als Suche mit einer Teilziel-Menge
- Hierarchische Planung: Planung in Dimension der Abstraktion