State-Space Planning und STRIPS

Hauptseminar Kognitive Systeme
WS 06/07

19. Januar 2006

Michael Räther

Inhalt

1. Definitionen
2. Repräsentation

   2.1 Mengen-theoretische Repräsentation
   2.2 Klassische Repräsentation

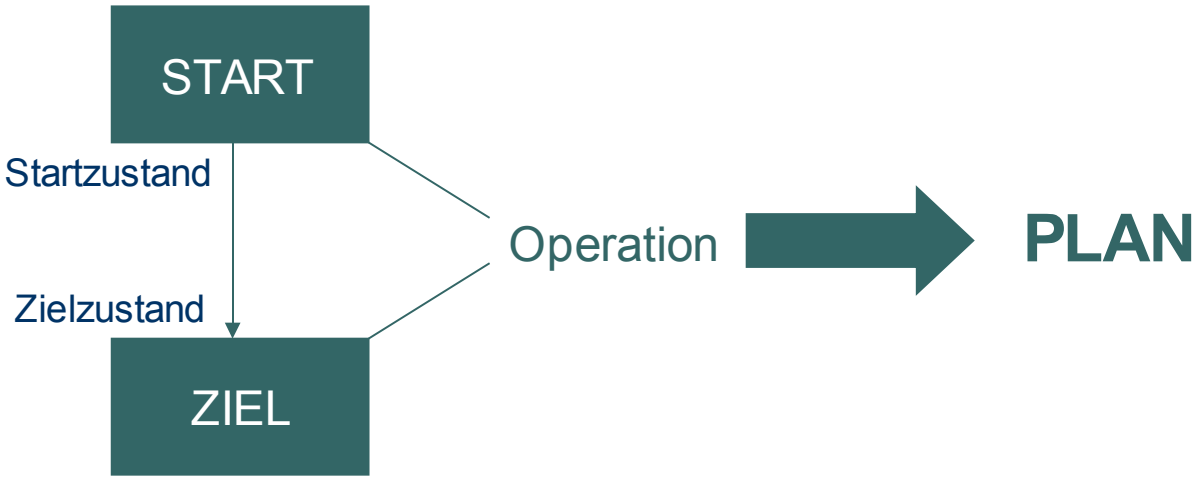
3. STRIPS
4. Diskussion und Ausblick

1. Definitionen (A)

Repräsentation:	Formale Darstellung der Welt
Zustand:	Formale Darstellung eines Zustands in der gegebenen Welt
Aktion/Operation:	dient zur Manipulation eines Zustands um somit weitere Zustände zu erschliessen
Zustandübergangs-funktion:	einmalig definierte Funktion, die den Gebrauch von Aktionen/Operationen definiert
Domäne:	Definition der Welt inklusive Operationen
Problem:	Problemstellung in der gegebenen Domäne (z.B. das Erreichen eines Zielzustandes in der gegebenen Problemdomäne)

1. Definitionen (B)

Plan: Lösung des gegebenen Problems in der gegebenen Problemdomäne
(Sequenz von Aktionen/Operationen die das gegebene Problem lösen)



1. Definitionen (C)

Zielsuche: Verwirklichung durch Suchalgorhitmen

Anwendung von Aktionen einer Domäne führt zu weiteren Zuständen auf die wiederum Aktionen ausgeführt werden können.
Wird ein Zustand erreicht, der gewisse Bedingungen erfüllt, so stellt die Sequenz von Aktionen die zu diesen Zustand geführt hat einen gültigen Plan für das Suchproblem dar.

Zwei grundsätzliche Vorgehensweisen:

- Vorwärtssuche
- Rückwärtssuche

Inhalt

1. Definitionen
2. Repräsentation
   - 2.1 Mengen-theoretische Repräsentation
     2.2 Klassische Repräsentation
3. STRIPS
4. Diskussion und Ausblick

2. Mengen-theoretische Repräsentation

Aufbau einer mengen-theoretischen Domäne:

- Grundlage ist die Faktenmenge L
- Menge an Zuständen S (jeder Zustand s ist eine Teilmenge von L)
- Menge an Aktionen A
- einmalig definierte Zustandübergangsfunktion y

2.1 Beispiel – Das Raketenproblem (A)

Symbolmenge (Fakten):

L = { onEarth, onMoon, atBaseEarth, atBaseMoon, loaded }

onEarth: Objekt befindet sich auf dem Planenten Erde
onMoon: Objekt befindet sich auf dem Mond

atBaseEarth: Rakete ist in der Erdbasis
atBaseMoon: Rakete ist in der Mondbasis

loaded: Objekt ist in der Rakete (Rakete ist beladen)

2.1 Beispiel – Das Raketenproblem (B)

Menge von Aktionen:

A = { move1, move2, load, unload }

move1 = ({atBaseEarth} , {atBaseEarth} , {atBaseMoon})
move2 = ({atBaseMoon} , {atBaseMoon} , {atBaseEarth})

load = ({atBaseEarth, onEarth} , {onEarth} , {loaded})
unload = ({atBaseMoon, loaded} , {loaded} , {onMoon})

2.1 Beispiel – Das Raketenproblem (C)

Initial-Zustand:

s_0 = { onEarth, atBaseMoon }

Endzustand (goal):

g = { onMoon, atBaseEarth }

2.1 Beispiel – Das Raketenproblem (D)

Planungsdomäne:

$\sum = (S, A, y)$

=> unabhängig von Start- und Endzustand

Planungsproblem:

$P = (\sum, s_0, g)$

=> macht Definition sämtlicher Zustände notwendig

Alternative:

P = (A, s0, g)

2.1 Beispiel – Das Raketenproblem (E)

Zustände (A):

s_0 = { onEarth, atBaseMoon }

move2 = ({atBaseMoon} , {atBaseMoon} , {atBaseEarth})

s_1 = { onEarth, atBaseMoon, atBaseEarth }

move1 = ({atBaseEarth} , {atBaseEarth} , {atBaseMoon})

load = ({atBaseEarth, onEarth} , {onEarth} , {loaded})

s_2 = { onEarth, atBaseEarth, loaded }

move1 = ({atBaseEarth} , {atBaseEarth} , {atBaseMoon})

s_3 = { atBaseEarth, loaded, atBaseMoon }

move2 = ({atBaseMoon} , {atBaseMoon} , {atBaseEarth})

unload = ({atBaseMoon, loaded} , {loaded} , {onMoon})

s_4 = { loaded, atBaseMoon, onMoon }

2.1 Beispiel – Das Raketenproblem (F)

Zustände (B):

s_4 = { atBaseMoon, onMoon }

move2 = ({atBaseMoon} , {atBaseMoon} , {atBaseEarth})

s_5 = { atBaseMoon, atBaseEarth, onMoon }

g = { onMoon, atBaseEarth }

=> Zielzustand erreicht

verwendete Sequenz von Aktionen:

π_1 = ‹ move2, load, move1, unload, move2 ›

Inhalt

1. Definitionen
2. Repräsentation

   2.1 Mengen-theoretische Repräsentation
   2.2 Klassische Repräsentation

3. STRIPS
4. Diskussion und Ausblick

2.2 Klassische Repräsentation

Aufbau einer klassischen Domäne:

- Grundlage ist eine Logik erster Ordnung L
- darin enthalten endliche Anzahl von Prädikat- und Konstantensymbole
- ein Zustand ist eine Menge von Literalen (negativierte und nicht-negativierte Atome aus L)
- einmalig definierte Zustandübergangsfunktion y

2.2 Beispiel – Das Raketenproblem (A)

Menge an konstanten Symbolen („Objekte“):

C = { r1, r2, p1, p2, o }

r1, p2 :	Transportmittel Rakete
p1, p2, m1 :	Planeten
o :	Objekt das transportiert werden soll

2.2 Beispiel – Das Raketenproblem (B)

Menge an Prädikatsymbolen (Relationen):

P = { at (r, p), loaded (r, o), free (p), unloaded (r),
adjacent (p, m), stored (p, o) }

at (r, p):	Rakete befindet sich auf Planeten
loaded (r, o):	Rakete ist mit Objekt beladen
free (p):	keine Rakete befindet sich auf dem Planeten
adjacent (p, m):	zwei Planeten sind benachbart (und somit erreichbar)
stored (p, o):	Planet hat Objekt gelagert

2.2 Relationen

Verschiedene Arten von Relationen:

- flexible Relationen (flexible relations)
  - Operatoren haben durch ihre Effekte Einfluss auf diese Relationen
    [z.B. at (r, p)]
- feststehende Relationen (rigid relations)
  - Relationen die sich durch Operatoren nicht ändern (können somit nie als Effekt eines Operators auftreten, jedoch eventuell als Voraussetzungen)
    [z.B. adjacent (p, m)]
    -> gilt jedoch nicht für gezeigtes Beispiel wegen buildStargate (p, m)

    => gelten somit für alle Zustände der Domäne

2.2 Beispiel – Das Raketenproblem (C)

Operatoren:

move (r, p, m)
;; Rakete fliegt von Planet p zum benachbarten Planet m
precond: adjacent (p, m), at (r, p), free (m)
effects: at (r, m), ¬ free (m), free (p), ¬ at (r, p),

load (r, o)
;; Rakete lädt Objekt
precond: at (r, p), stored (p), unloaded (r)
effects: ¬ stored (p), loaded (r, o), ¬ unloaded (r)

unload (r, o)
;; Rakete entlädt Objekt
precond: at (r, p), loaded (r, o)
effects: stored (p), ¬ loaded (r, o), unloaded (r)

2.2 Beispiel – Das Raketenproblem (D)

Operatoren:

buildStargate (p, m)
;; Planet baut Stargate zu anderen Planeten
precond: stored (p, o)
effects: adjacent (p, m)

2.2 Objekt-Instantiierung (A)

Beispiel: Instantiierung eines Operators

move (r, p, m)
;; Rakete fliegt von Planet p zum benachbarten Planet m
precond: adjacent (p, m), at (r, p), free (m)
effects: at (r, m), ¬ free (m), free (p), ¬ at (r, p),

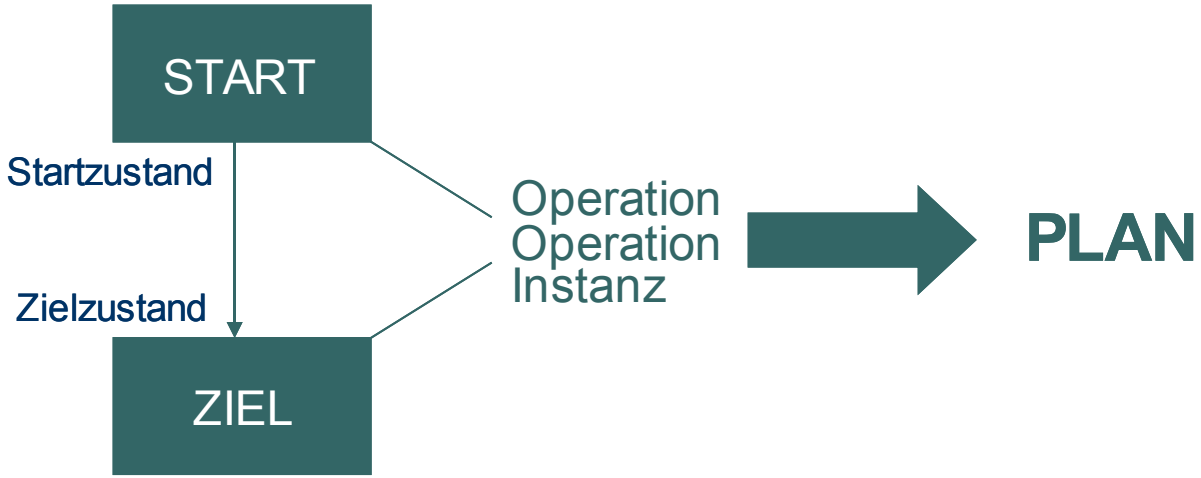
C = { r1, r2, p1, p2, o }

move (r1, p1, p2)
;; Rakete r2 fliegt von Planet p1 zum benachbarten Planeten p2
precond: adjacent (p1, p2), at (r1, p1), free (p2)
effects: at (r1, p2), ¬ free (p2), free (p1), ¬ at (r1, p1),

Vorteil: - ermöglicht variablen Einsatz
(könnte z.B. auch mit r2 instantiiert werden)

=> kompakte(re) Repräsentation möglich

2.2 Objekt-Instantiierung (B)



2.2 Beispiel – Das Raketenproblem (E)

Initial-Zustand:

s_0 = { at(r1, p1), at(r2, p2), ¬ free(p1), ¬ free(p2), free(m1), loaded(r1, o), ¬ loaded(r2, o), ¬ stored(p1, o), ¬ stored(p2, o), adjacent(p1, p2), adjacent(p2, m1) }

Endzustand (goal):

g = { stored(p2, o), ¬ at(r2, m1) }

2.2 Beispiel – Das Raketenproblem (F)

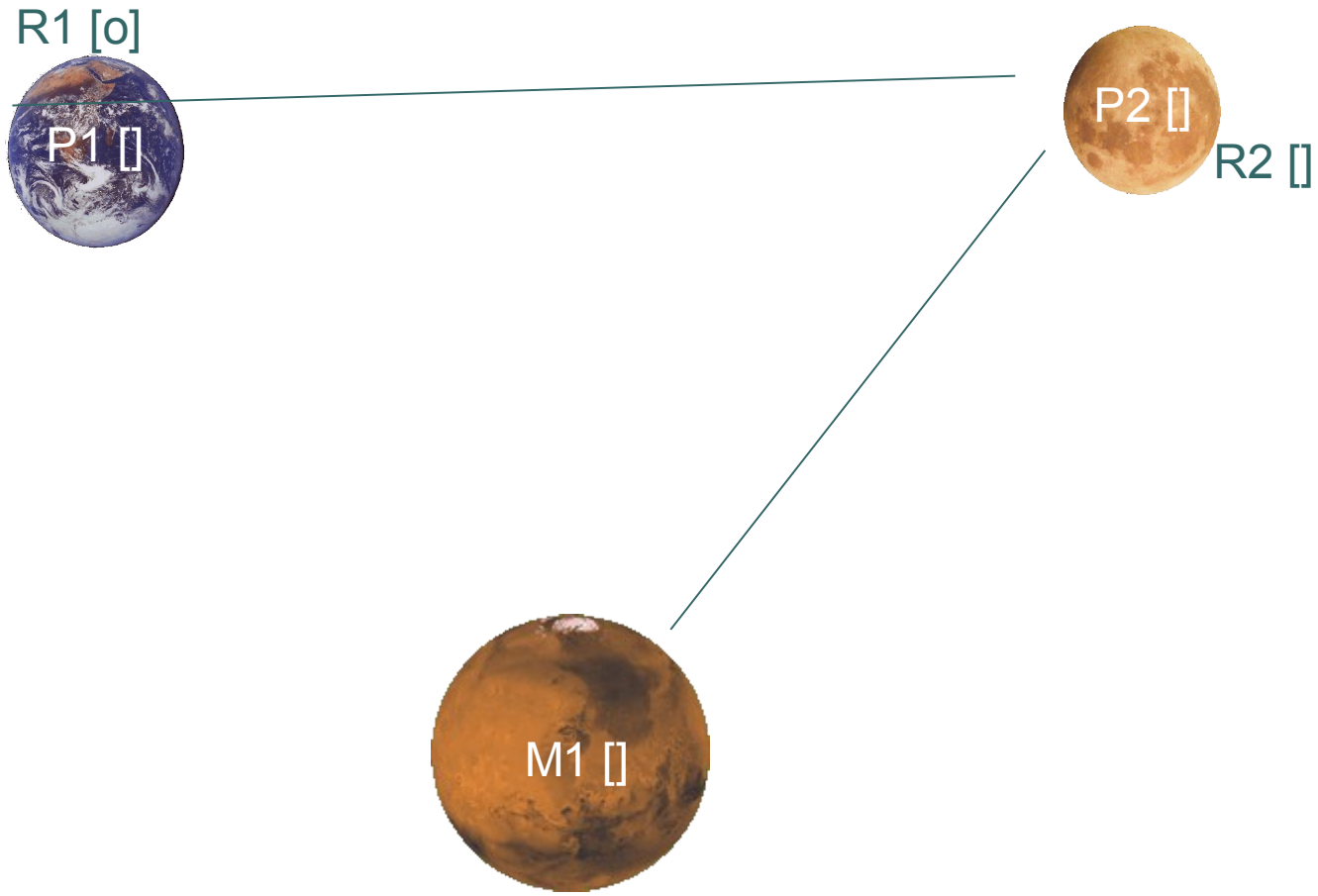
Initial-Zustand (in Worten):

Rakete1 ist mit Objekt beladen und befindet sich auf Planet p1.
Rakete2 ist unbeladen und befindet sich auf Planet p2.
Die Planeten p1 und p2 sowie p2 und m1 sind zueinander benachbart.

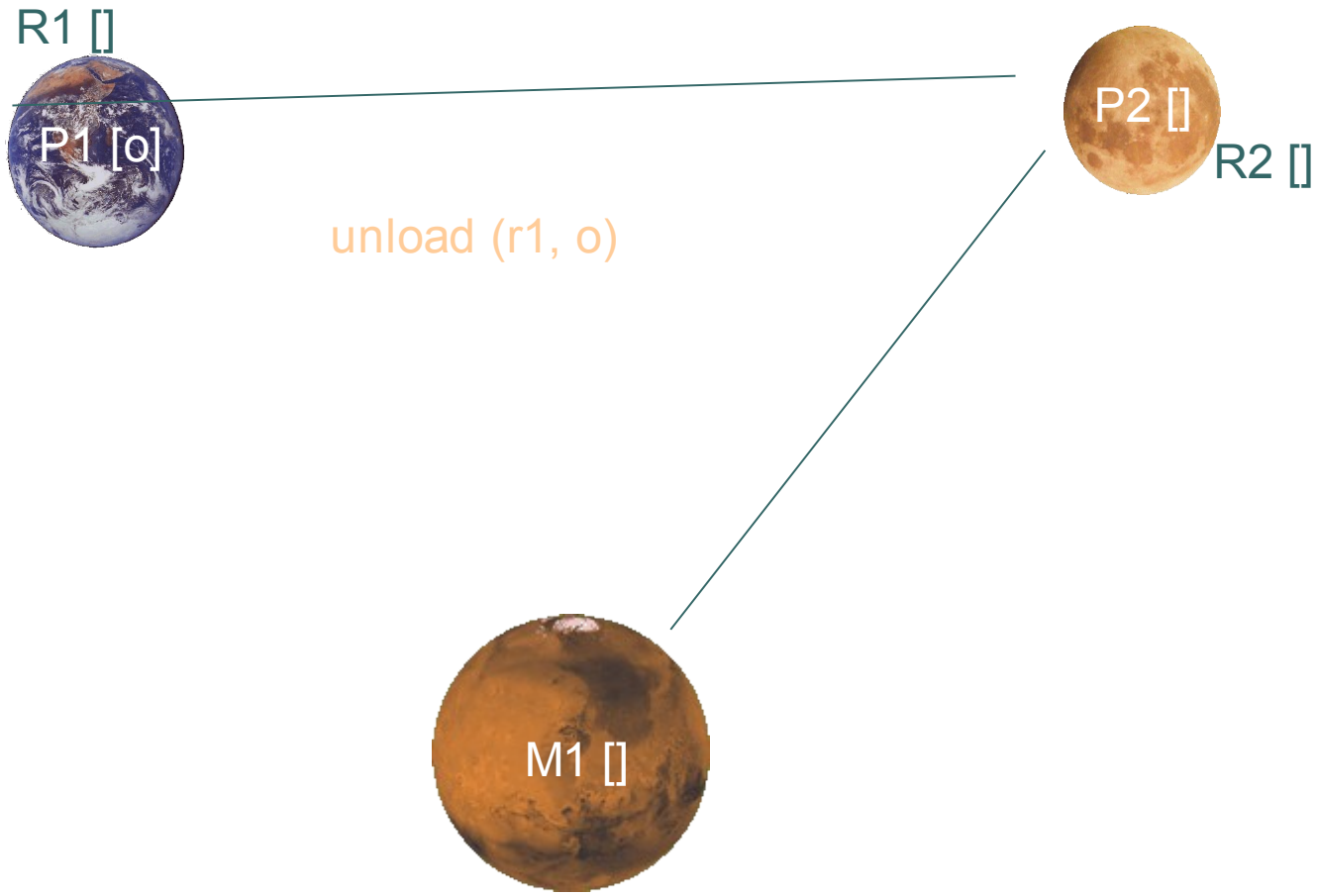
Endzustand (in Worten):

Rakete1 soll sich auf Planeten m1 befinden.

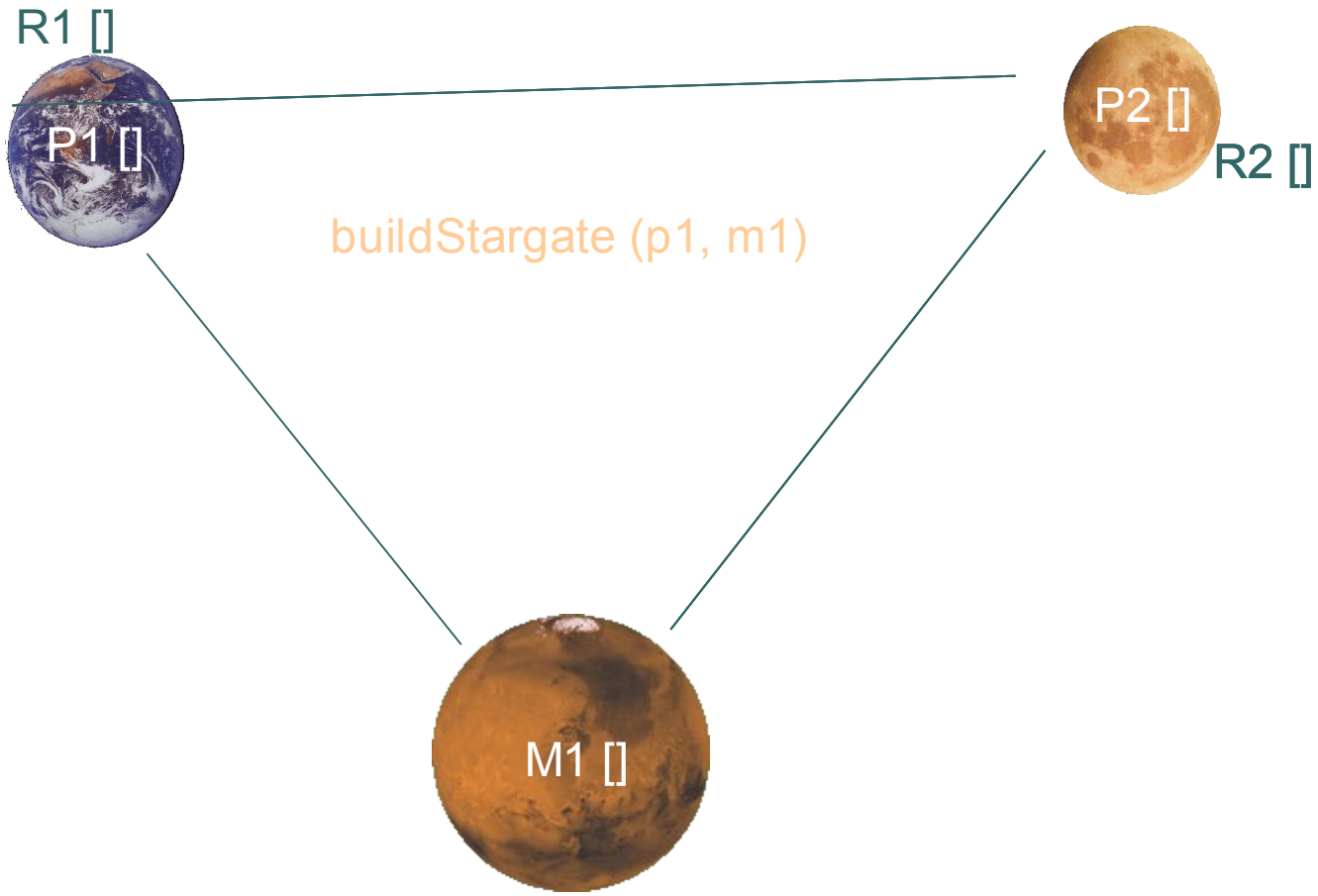
2.2 Beispiel – Das Raketenproblem Minimallösung (A)



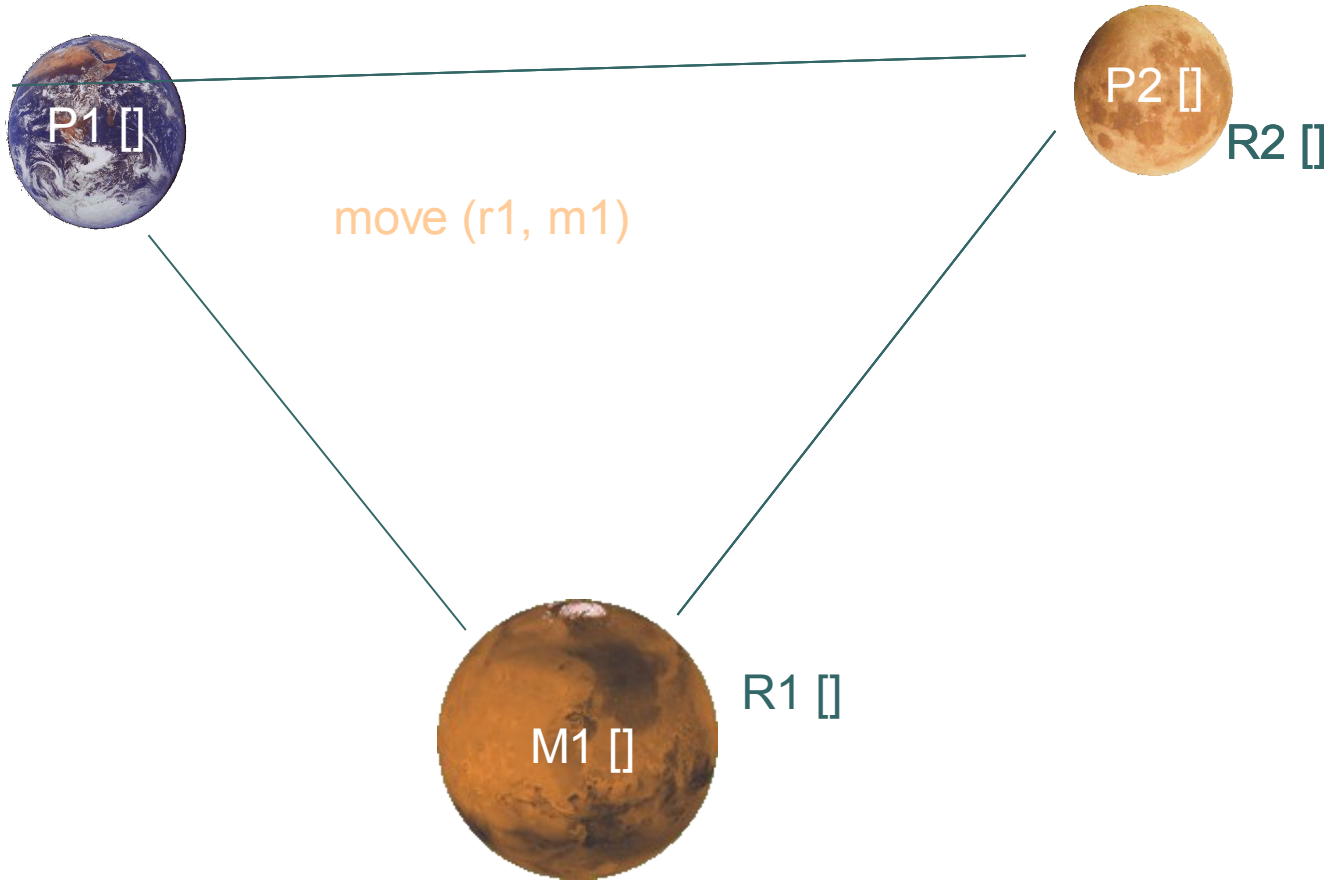
2.2 Beispiel – Das Raketenproblem Minimallösung (B)



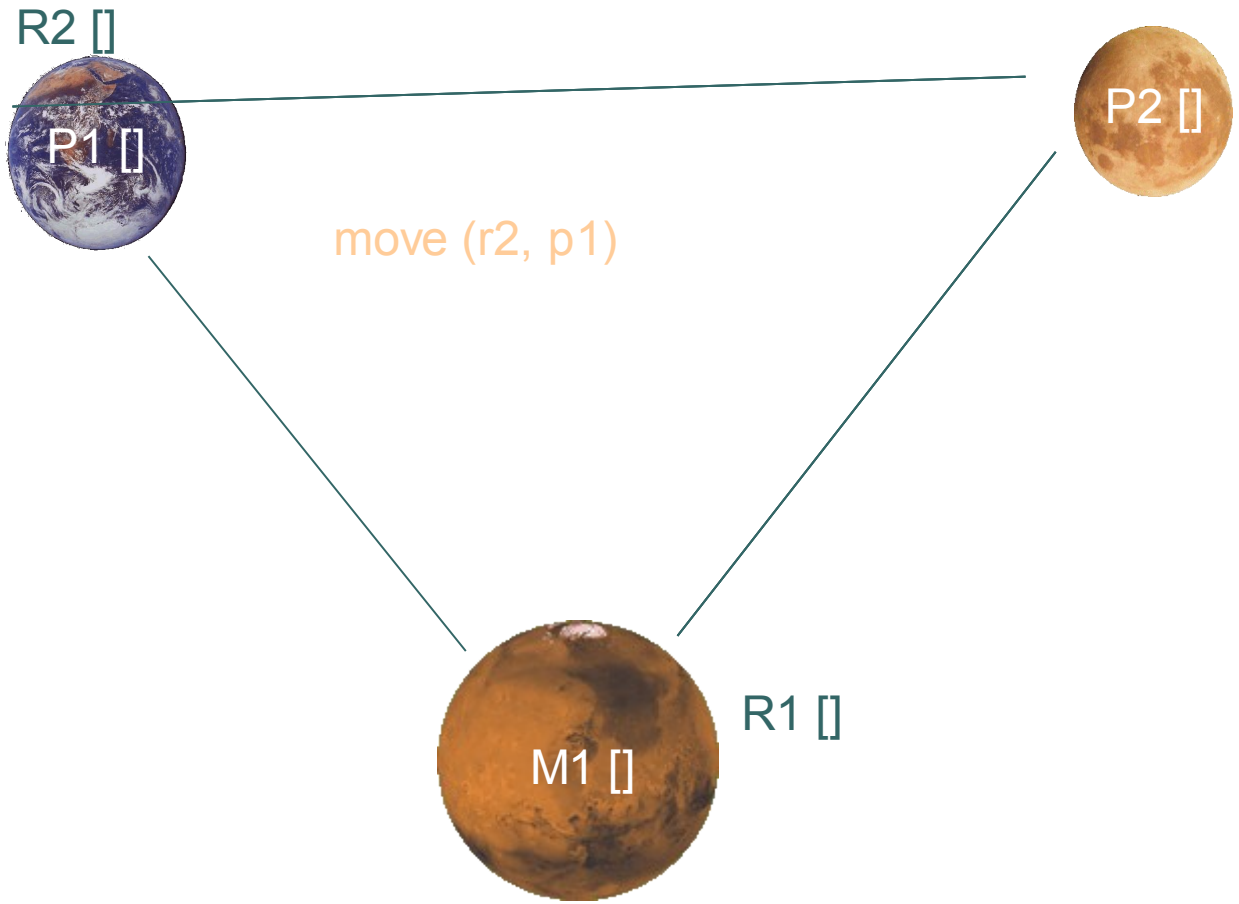
2.2 Beispiel – Das Raketenproblem Minimallösung (D)



2.2 Beispiel – Das Raketenproblem Minimallösung (E)



2.2 Beispiel – Das Raketenproblem Minimallösung (F)



2.2 Beispiel – Das Raketenproblem Minimallösung (G)

Minimallösung:

π_1 = ‹ unload(r1, o), buildStargate(p1, m1),
move(r1, m1), move(r2, p1) ›

2.2 Erweiterungen der klassischen Repräsentation

Mögliche Erweiterungen:

- konditionale Operatoren

- quantifizierte Ausdrücke

- disjunktive Voraussetzungen

Inhalt

1. Definitionen
2. Repräsentation

   2.1 Mengen-theoretische Repräsentation
   2.2 Klassische Repräsentation

3. **STRIPS**
4. Diskussion und Ausblick

3. STRIPS

STanford **R**esearch **I**nstitute **P**roblem **S**olver

- Grundlage ist die klassische Repräsentation
- 1971 entwickelt von Fikes und Nilson als linearer Planer
- später Erweiterung zu formaler Sprache
- heutiges Einsatzgebiet: Forschung (Entwicklung und Test von neuen Algorhitmen)

3. STRIPS Einschränkungen

- nur positive Literale erlaubt (z.B. ¬ at(r1, p1) nicht erlaubt!)
- Operatoren besitzen neben Voraussetzungen eine Add- und Delete-Liste
- Add-Liste: Literale die hinzugefügt werden
- Delete-Liste: Literale die gelöscht werden

3. STRIPS Vorwärtssuche (Progression)

- Startzustand als Ausgangspunkt für den Algorithmus
- sinnvoll bei überschaubaren Problemen mit kleinen Zustandsraum
- bei komplexeren Problemen ist Bewertung nötig (welcher Operator ist sinnvoll = Heuristik)

```
Forward-search (O, s0, g)
        s ← s0
        π ← the empty plan
        loop
                if s satisfies g then return π
                applicable ← { a | a is a ground instance of an operator
                                   in O, and precond(a) is true in s }
                if applicable = emptySet then return failure
                nondeterministically choose an action a isElem applicable
                s ← y(s,a)
                π ← π.a
```

3. STRIPS Rückwärtssuche (Regression)

- Endzustand als Ausgangspunkt für den Algorithmus
- Suchbaum kleiner und somit schnellere Suche
- liefert eventuell inkonsistente Zustände

```
Backward-search (O, s0, g)
        π ← the empty plan
        loop
                if s0 satisfies g then return π
                relevant ← { a | a is a ground instance of an operator
                                 in O that is relevant for g }
                if relevant = emptySet then return failure
                nondeterministically choose an action a isElem applicable
                π ← a.π
                g ← y^-1(g,a)
```

3. STRIPS Algorithmus (A)

- ähnelt dem Regressionsalgorithmus
- in jeder Rekursion werden nur Teilziele verfolgt, die durch die Voraussetzungen (Preconditions) des letzten ausgeführten Operators bestimmt werden
- erfüllt ein Zustand alle Voraussetzungen eines Operators, so wird dieser Operator ausgeführt und diese Bindung wird nicht wieder gelöst

=> Suchraum wird bedeutend kleiner; STRIPS wird dadurch jedoch auch unvollständig

Beispiel: Vertauschung der Wertzuweisungen zweier Variablen;
STRIPS findet hier keine Lösung!

3. STRIPS Algorithmus (B)

```
Ground-STRIPS (O, s0, g)
        π ← the empty plan
        loop
                if s0 satisfies g then return π
                A ← { a | a is a ground instance of an operator
                          in O, and a is relevant for g }
                if A = emptySet then return failure
                nondeterministically choose an action a isElem A
                π‘ ← Ground-STRIPS (O, s, precond(a))
                if π‘ = failure then return failure
                ;; if we get here, then π‘ achieves precond(a) from s
                s ← y(s, π‘)
                ;; s now satisfies precond(a)
                s ← y(s,a)
                π ← π. π‘.a
```

Inhalt

1. Definitionen
2. Repräsentation

   2.1 Mengen-theoretische Repräsentation
   2.2 Klassische Repräsentation

3. STRIPS
4. **Diskussion und Ausblick**