Kapitel MK:VI

III. Planning

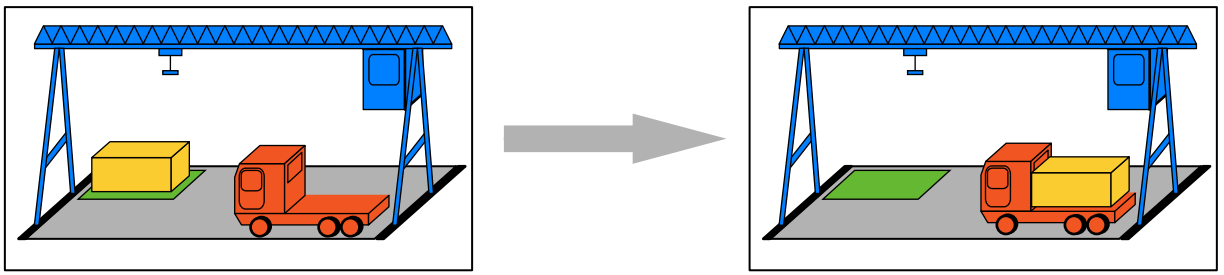
- Motivation
- Wissensrepräsentation
- Planungsalgorithmen
- Suche im Zustandsraum
- Suche im Planraum
- Komplexität
- Erweiterungen

Bemerkungen:

- Dieses Kapitel orientiert sich an:
  - Malik Ghallab, Dana Nau, Paolo Traverso. *Automated Planning: Theory and Practice* (Lecture Slides)
  - David Poole, Alam Mackworth, Randy Goebel. *Computational Intelligence: A Logical Approach*
    Kap. 8 (Lecture Slides)
  - Stuart Russell, Peter Norvig. *Artificial Intelligence: A Modern Approach*
    Kap. 11 (Lecture Slides)
  - Nils Nilsson. *Principles of Artificial Intelligence*
    Kap. 7, 8
  - Jacques Ferber. *Multi-Agent Systems: An Introduction to Artificial Intelligence*
    Kap. 7, 8

Motivation

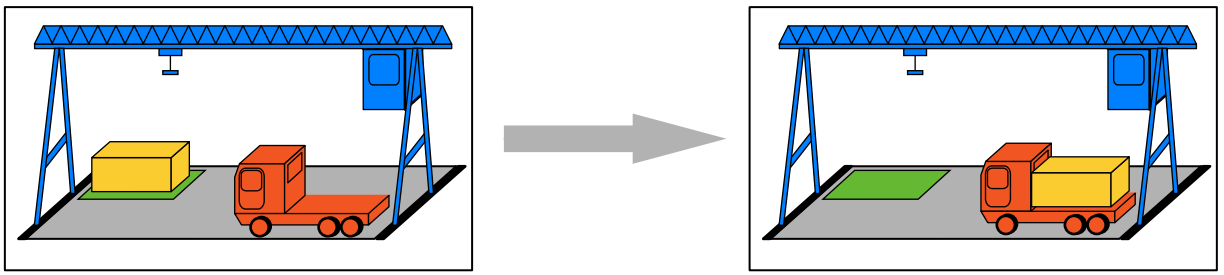
Beispiel: Verladestation (Ghallab, Nau, Traverso)



➜ Wie kann der LKW beladen werden?

Motivation

Beispiel: Verladestation (Ghallab, Nau, Traverso)



➜ Wie kann der LKW beladen werden?

Abstraktion von konkreten Problemstellungen:

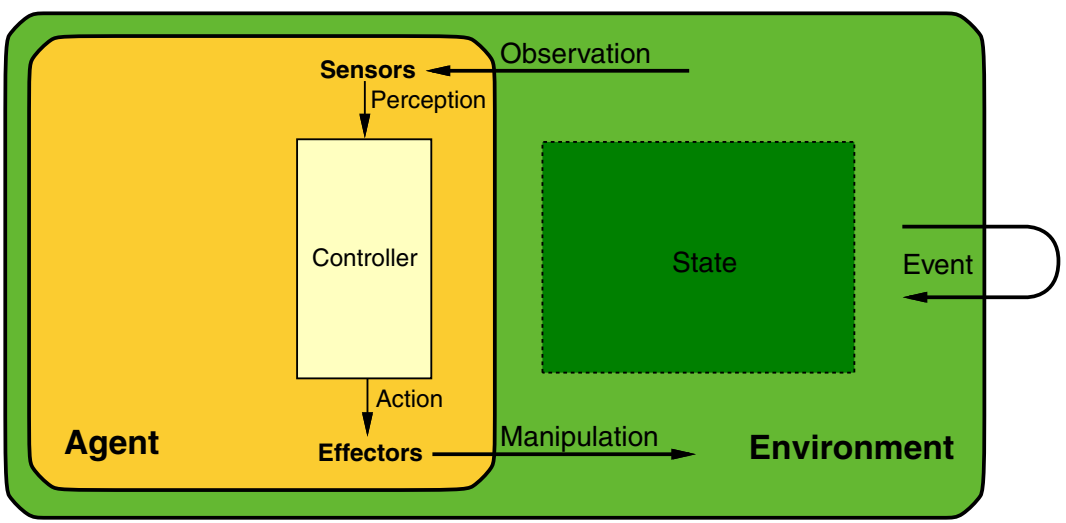
- Zusammenfassung von möglichen Standorten zu abzählbarer/endlicher Menge von Konfigurationen für Ladung, LKW, . . .
- Zusammenfassung von Veränderungen zu abzählbarer/endlicher Menge von Aktionen und Events

➜ Problemlösung durch Suche nach einer passenden Aktionenfolge

Motivation

Einfaches formales Modell $(\mathcal{A}, \mathcal{S})$ von Agentensystemen (mit Events)

- Umgebung modelliert als State Transition System (STS-E) $\mathcal{S} = (S, A, E, \tau)$
  - endliche/abzählbare Menge von Zuständen S
  - endliche/abzählbare Menge von Aktionen A
  - endliche/abzählbare Menge von Events E
  - Übergangsrelation $\tau \subseteq (S \times A \times E) \times S$
- Agent $\mathcal{A} = ...$

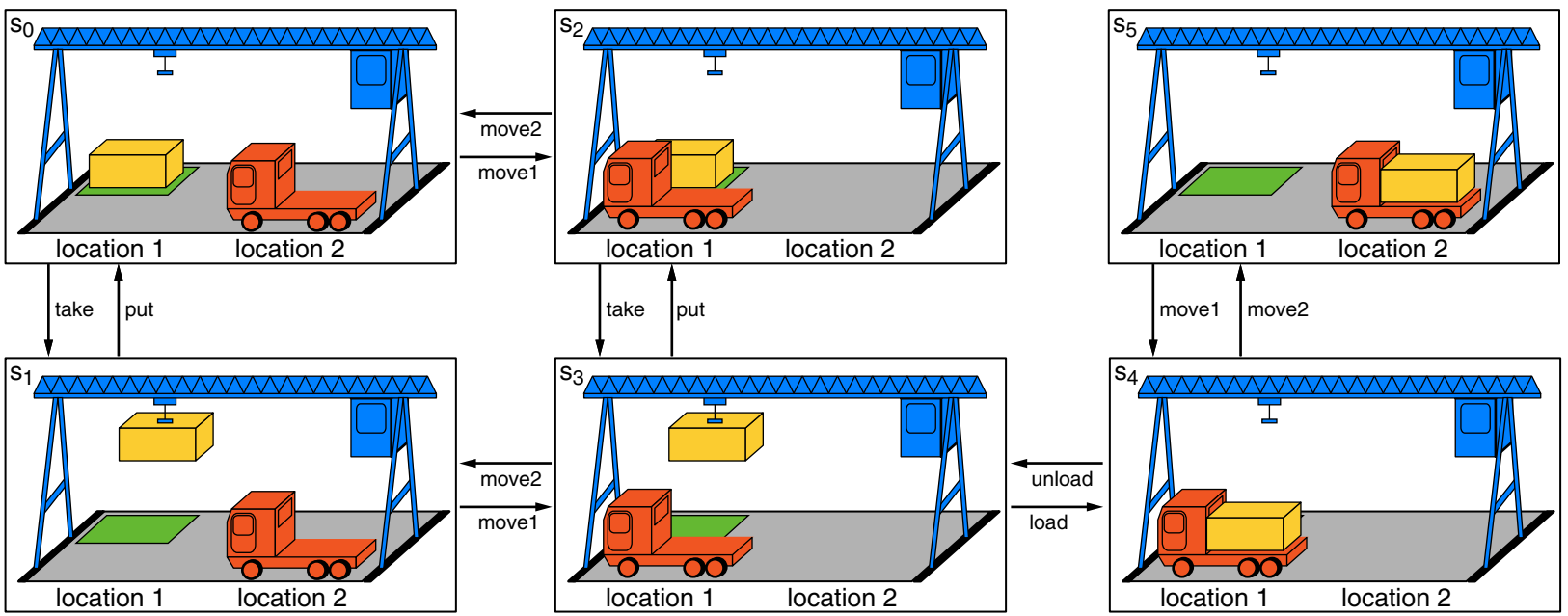


Motivation

Beispiel: Verladestation (Fortsetzung)

Formales Modell der Umgebung $\mathcal{S}$

- Zustände $S = \{s_0, \ldots, s_5\}$
- Aktionen $A = \{\mathit{move1}, \mathit{move2}, \mathit{put}, \mathit{take}, \mathit{load}, \mathit{unload}\}$
- Events $E = \emptyset$
- Übergangsrelation τ siehe Graphik

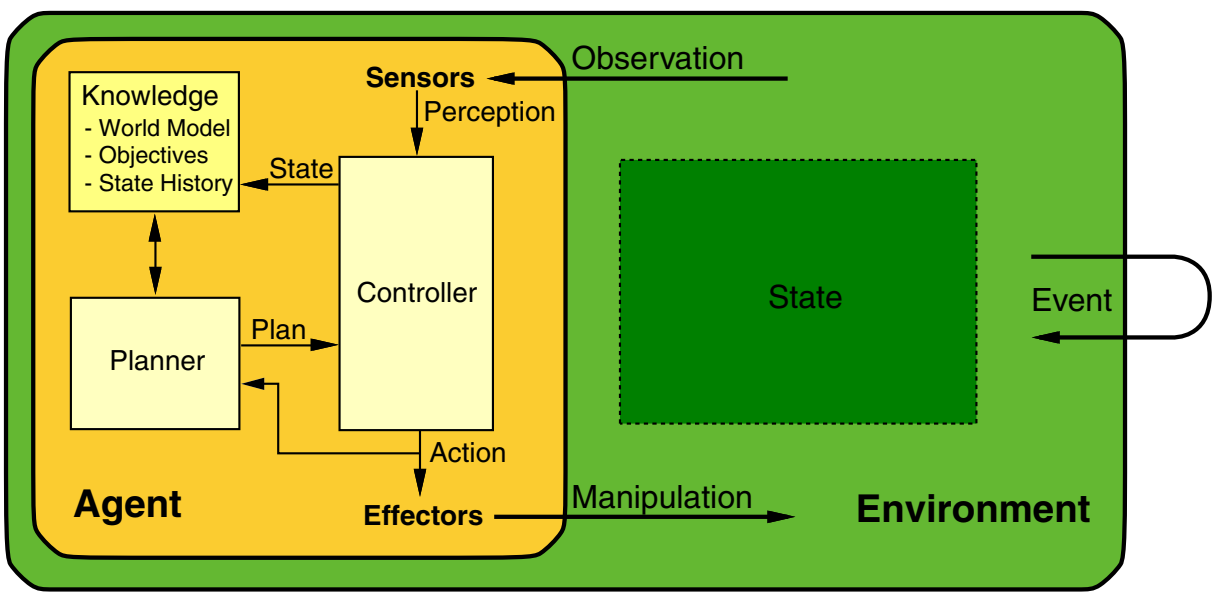


Motivation

(Einfaches) Planungsproblem für Agenten

- Ein Zustandsübergangssystem / State Transition System $\mathcal{S} = (S, A, E, \tau)$ für die Umgebung sei gegeben.
- s_{init} Anfangszustand, $s_{init} \in S$
- s_{final} Zielzustand, $s_{final} \in S$

➜ Gesucht ist ein Plan $p = (a_1, \ldots, a_n)$, d.h. eine endliche Folge von Aktionen aus A mit $s_{init} \xrightarrow{a_1} \ldots \xrightarrow{a_n} s_{final}$



Bemerkungen:

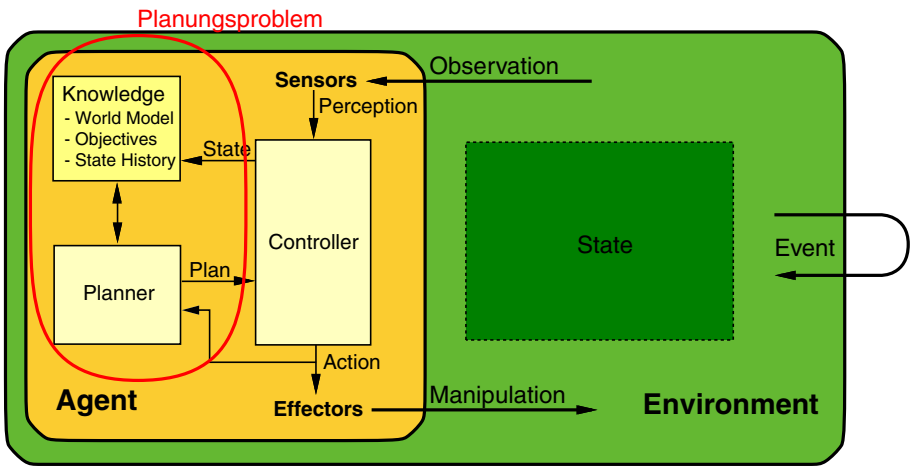
- Anstelle eines einzelnen Zielzustandes kann in einem Planungsproblem auch gefordert werden,
  - einen Zielzustand aus einer Menge von Zielzuständen zu erreichen,
  - eine Menge von Zielzuständen in einer festen Reihenfolge nacheinander zu erreichen,
  - eine Menge von Aufgaben zu erfüllen,
  - eine Zielfunktion zu maximieren,
  - ...
- Für Komplexitätsuntersuchungen schränkt man die Aufgabe häufig ein:
  Existiert ein Plan (der Länge k) mit den geforderten Eigenschaften?

Bemerkungen: (Fortsetzung)

Das einfache formale Modell legt nahe, dass wir nur eine einfache Form von Umgebungen betrachten wollen, nämlich Umgebungen, die ein Zustandsübergangssystem mit Events bilden. Eine andere Sichtweise ist hilfreicher:

Durch seine Sensoren nimmt ein Agent seine Umgebung wahr. Die Sensoren sind im Vergleich zur realen Umgebung ungenau, so dass der tatsächliche Zustand für den Agenten gar nicht vollständig beobachtbar sein kann. Die vergröberte Sicht wird agentenintern noch weiter vereinfacht, um sie handhabbar zu machen. Agentenintern arbeitet man mit einer Abstraktion der Umgebung in Form eines Zustandsübergangssystems.

Das einfache formale Modell beschreibt die agenteninterne Vorstellung über das Funktionieren der Umgebung.



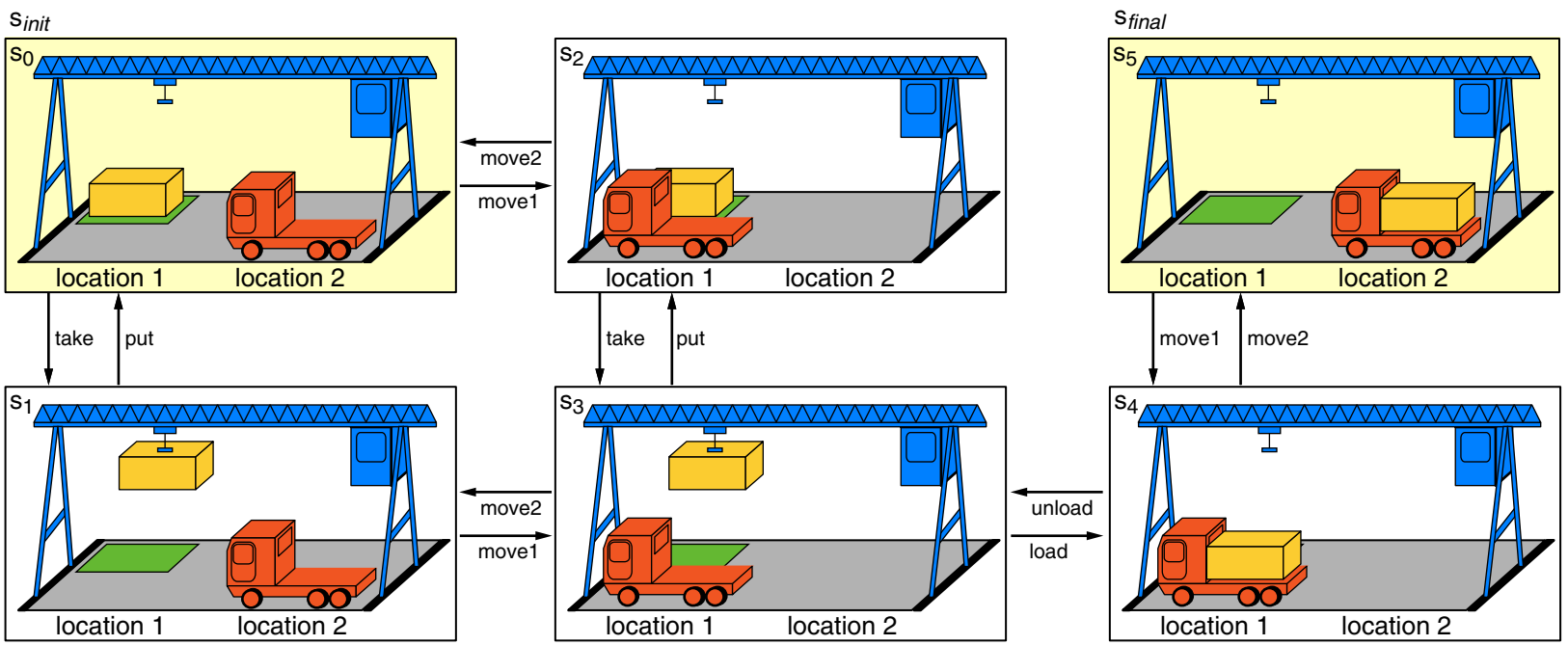
Die Planungsaufgabe wird formuliert in Termini des formalen Modells, das ein Agent von seiner Umgebung hat. Man kann sie als agenteninterne Übersetzung einer Planungsaufgabe auffassen, die an das reale System aus Agent und Umwelt gestellt wurde.

Motivation

Beispiel: Verladestation (Fortsetzung)

Zustandsübergangsmodell STS-E $\mathcal{S} = (S, A, E, \tau)$

- Zustände $S = \{s_0, \ldots, s_5\}$
- Aktionen $A = \{\mathit{move1}, \mathit{move2}, \mathit{move1}, \mathit{put}, \mathit{take}, \mathit{load}, \mathit{unload}\}$
- Events $E = \emptyset$
- Übergangsrelation τ siehe Graphik



Motivation

Plan vs. Policy

- Plan für die Verladeaufgabe:
  $p = (\textit{take}, \textit{move1}, \textit{load}, \textit{move2})$
- Policy $\pi : S \rightarrow A$ für den Verladeagenten:
  $\pi = \{(s_0, \textit{take}), (s_1, \textit{move1}), (s_3, \textit{load}), (s4, \textit{move2})\}$

Worin besteht der Unterschied?

Motivation
Mögliche Einschränkungen des Planungsproblems

A0 Endliches System
Die Menge der Zustände S ist endlich.

A1 Vollständige Wahrnehmbarkeit
Der Zustand des Systems kann vollständig wahrgenommen werden, es besteht keine Unsicherheit über den aktuellen Zustand.

A2 Deterministisches System
Jede Aktion hat in jedem Zustand genau einen Folgezustand: $\tau : S \times A \times E \to S$.

A3 Statisches System
Es gibt keine exogenen Events, Zustandsänderungen geschehen nur aufgrund von Aktionen des Agenten: $E = \emptyset$.

A4 Eingeschränkte Ziele
Eine Menge von möglichen Zielzuständen ist vorgegeben.

A5 Sequentielle Pläne
Pläne sind linear geordnete Folgen von Aktionen.

A6 Implizite Zeit
Es gibt keine Zeitdauern, sondern nur eine Folge von unmittelbar eintretenden Zuständen.

A7 Offline Planning
Der Planer berücksichtigt keine Zustandsänderungen während des Planens, die auf Events zurückzuführen sind.

➜ Bedingungen A0-A7: Klassisches Planen

Bemerkungen:

- Wenn nach A0 die Menge der Zustände endlich ist, nach A2 die Relation τ eine Funktion ist und nach A3 die Menge E leer ist, dann macht ein unendliche Mengen von Aktionen keinen Sinn. Man kann die Aktionen, die für alle Zustände zu gleichen Folgezuständen führen, zu Äquivalenzklassen zusammenfassen. Von denen gibt es aber nur endlich viele.

Motivation
Mögliche Einschränkungen des Planungsproblems

A0 Endliches System
Die Anzahl von Zuständen, Aktionen und Events ist endlich.

A1 Vollständige Wahrnehmbarkeit
Der Zustand des Systems kann vollständig wahrgenommen werden.

A2 Deterministisches System
Jede Aktion hat in jedem Zustand genau einen Folgezustand.

A3 Statisches System
Es gibt keine exogenen Events, Zustandsänderungen geschehen nur aufgrund von Aktionen des Agenten.

A4 Eingeschränkte Ziele
Eine Menge von möglichen Zielzuständen ist vorgegeben.

A5 Sequentielle Pläne
Pläne sind linear geordnete Folgen von Aktionen.

A6 Implizite Zeit
Es gibt keine Zeitdauern, sondern nur eine Folge von unmittelbar eintretenden Zuständen.

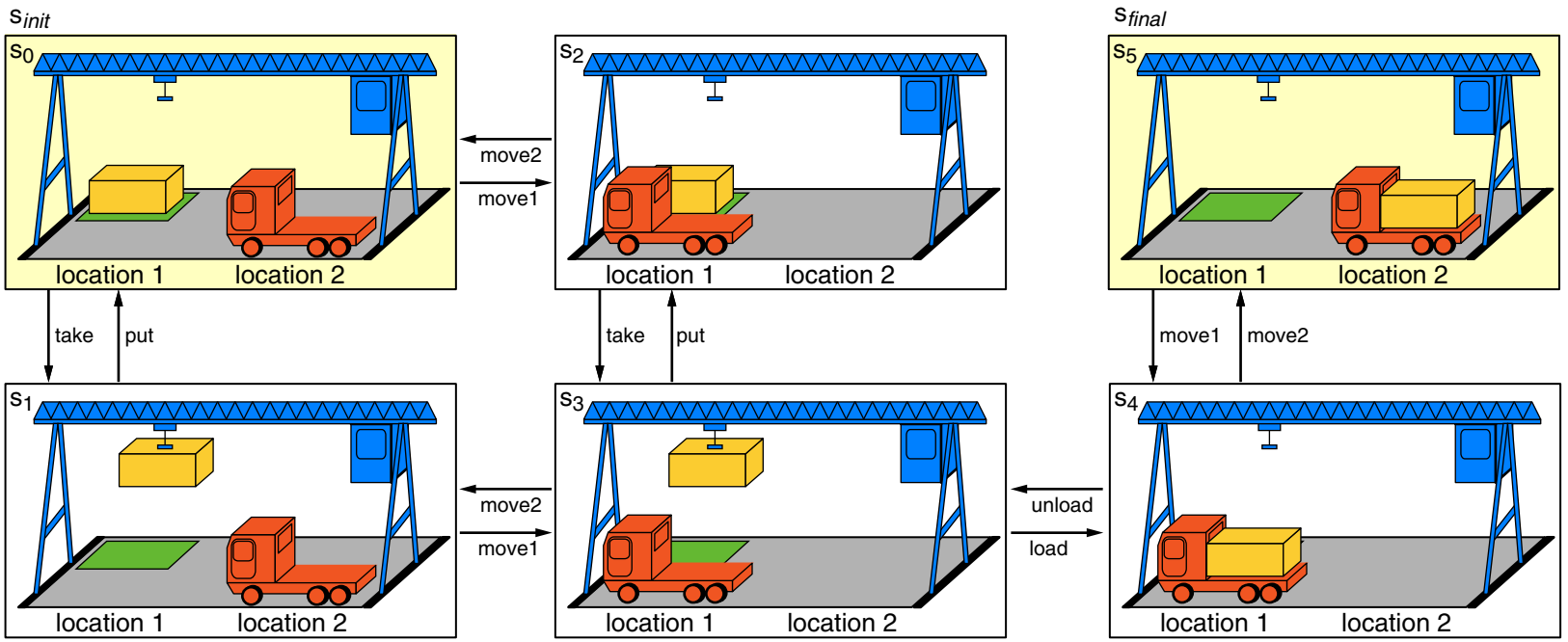
A7 Offline Planning
Der Planer berücksichtigt keine Zustandsänderungen während des Planens, die auf Events zurückzuführen sind.

➜ Bedingungen A1-A7: Planen kann beschrieben werden als Suche nach einem Pfad im Raum der Zustände mit Aktionen als Kanten.

Wissensrepräsentation

Beispiel: Verladestation (Fortsetzung)

Zustände $S = \{s_0, \ldots, s_5\}$



➜ Wie beschreibt man *große* Zustandsmengen?

Wissensrepräsentation

Beschreibung eines Zustandes

Ansatz:

- Beschreibung durch Angabe von Eigenschaften

Ziel:

- Eindeutige Beschreibung für jeden Zustand
- Einfache Semantik
- Unabhängigkeit vom konkreten Anwendungsfall

Wissensrepräsentation

Beschreibung eines Zustandes

Ansatz:

- Beschreibung durch Angabe von Eigenschaften

Ziel:

- Eindeutige Beschreibung für jeden Zustand
- Einfache Semantik
- Unabhängigkeit vom konkreten Anwendungsfall

➜ Verwendung von logikbasierten Beschreibungen

- Aussagenlogik: *block1_liegt_auf_platz1*
- Prädikatenlogik: $liegt\_auf(block1, platz1)$

Wissensrepräsentation

Beschreibung eines Zustandes

Ansatz:

- Beschreibung durch Angabe von Eigenschaften

Ziel:

- Eindeutige Beschreibung für jeden Zustand
- Einfache Semantik
- Unabhängigkeit vom konkreten Anwendungsfall

➜ Verwendung von logikbasierten Beschreibungen

- Aussagenlogik: *block1_liegt_auf_platz1*
- Prädikatenlogik: $\mathit{liegt\_auf}(\mathit{block1}, \mathit{platz1})$

➜ Wie beschreibt man *große* Zustandsmengen?

Wissensrepräsentation

Beschreibung eines Zustandsraumes

Ansatz:

- Explizite Beschreibung des Anfangszustandes
- Angabe von Operatoren für Zustandsübergänge

Ziel:

- Beschreibung der Anwendbarkeit von Aktionen
- Beschreibung der Zustandsveränderung durch Anwendung von Aktionen
- Generische Beschreibung durch Operatoren anstelle konkreter Aktionen

➜ Der benötigte Teil des Zustandsraumes kann je nach Bedarf durch Anwendung von Operatoren auf den Anfangszustand erzeugt werden.

Wissensrepräsentation

STRIPS Planning Language

(**ST**anford **R**esearch **I**nstitute **P**roblem **S**olver)

Sprache zur Beschreibung von Planungsproblemen:

- entwickelt von Fikes und Nilsson 1971,
- angewendet in *Shakey*,
  (Mobiler Roboter des Stanford Research Institute (SRI)
  mit Integration von Perzeption, Planung und Ausführung),
- entstanden im Spannungsfeld von
  - Suche im Zustandsraum,
  - Theorembeweisen und
  - Steuerungstheorie (Regelungstechnik).
- Planen mit aussagenlogischen STRIPS Modellen ist PSPACE-vollständig
  (Bylander 1991).

Bemerkungen:

- Die STRIPS Sprache ist als eine Möglichkeit zur Spezifikation von Planungsproblemen in die Sprache PDDL (**P**lanning **D**omain **D**escription **L**anguage) eingegangen.

Wissensrepräsentation

STRIPS Sprache (stark eingeschränkte Version)

- *Basis* der STRIPS Sprache ist Sprache der Prädikatenlogik erster Stufe (evtl. mit Gleichheit und Sorten) mit folgenden Einschränkungen:
  - nur endlich viele Prädikatssymbole,
  - nur endlich viele Konstantensymbole,
  - keine Funktionssymbole.

  ➜ *Unique Name Assumption (UNA):*
  Die Konstanten bezeichnen verschiedene Objekte.

- *Zustände* sind (als Konjunktion aufzufassende) endliche Mengen von atomaren variablenfreien Formeln.

  ➜ *Closed World Assumption (CWA):*
  Alle in einem Zustand enthaltenen Prädikate werden als wahr in dem Zustand angenommen, die nicht enthaltenen Prädikate als falsch.

Bemerkungen:

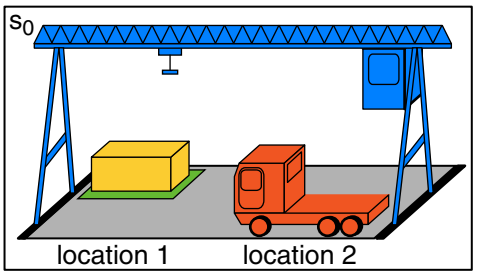
- In einer STRIPS Sprache sind als Terme nur die Konstanten und die Variablen möglich.
- In Zuständen erlaubte Formeln sind nur nicht-negierte Prädikate mit Konstanten als Argumenten ($\rightarrow$ Aussagenlogik). Die Menge aller solcher Formeln ist bereits aufgrund der Einschränkungen der Sprache endlich.
- *Domain Closure Assumption (DCA):*
  Alle Objekte der Welt sind explizit bennant. Es gibt keine weiteren Objekte.

  Diese Annahme liegt der Modellierung in der STRIPS Sprache ebenfalls zu Grunde.

Wissensrepräsentation

Beispiel: Verladestation (Fortsetzung)

- Konstanten, z.B.
  - *truck1* für Fahrzeug,
  - *location1* und *location2* für Fahrzeugpositionen,
  - *c1* für Container,
  - *pile1* für Stapelposition,
  - *crane1* für Portalkran.
- Prädikate, z.B.
  - $at(.,.)$ für Fahrzeugposition,
  - $empty(.)$ für freien Kran,
  - $unloaded(.)$ für unbeladenen Truck,
  - $top(.,.)$ für obersten Container auf Stapel,
  - $in(.,.)$ für Container auf Stapel.
- Zustand, z.B.
  $\{\ at(truck1, location2), unloaded(truck1),$
  $in(c1, pile1), top(c1, pile1), empty(crane1)\ \}$



Wissensrepräsentation

STRIPS Sprache (stark eingeschränkte Version) (Fortsetzung)

- *Operatoren* werden definiert durch zwei Mengen (Konjunktionen) von Literalen (negierten und nicht-negierten atomaren Formeln),
  - die Vorbedingungen *precond* und
  - die Effekte *effects*.
- Operatoren werden benannt durch $op(x_1, \ldots, x_n)$ mit einem eindeutigen Namen *op* und den verschiedenen in den Vorbedingungen und den Effekten auftretenden Variablen $x_1, \ldots, x_n$ als Argumente von *op*.
- Grundinstanzen von Operatoren beschreiben *Aktionen*.

Bemerkungen:

- In den Effekten *effects* von Operatoren können Variablen auftreten, die in den Vorbedingungen *precond* nicht vorkommen. Da Aktionen Grundinstanzen von Operatoren sind, werden auch diese Variablen instantiiert.
  Sucht man die in einem Zustand anwendbaren Aktionen, konzentriert man sich auf den *precond* Teil der Operatoren. Variablen, die nur im *effects* Teil vorkommen, könnte man dann als allquantifiziert auffassen wollen und alle möglichen Instanzen der zugehörigen Literale entfernen bzw. hinzufügen wollen. Dies ist NICHT zulässig.
- Die Vorbedingungen *precond* von Operatoren sollten keine atomare Formel negiert und gleichzeitig nicht-negiert enthalten, denn solch eine Vorbedingung kann von keiner Zustandsbeschreibung erfüllt werden. Diese Forderung gilt im Prinzip auch für die Aktionen. Die Grundsubstitutionen, die von Operatoren zu Aktionen führen, sollten keine komplementären Literale bewirken. Zugunsten einfacherer Operatormengen wird die zweite Forderung jedoch nicht immer erfüllt.
- In den Effekten *effects* von Operatoren sollte ebenfalls keine atomare Formel negiert und gleichzeitig nicht-negiert enthalten sein. Für eine Aktion, die Literale in beiden Polaritäten in den Effekten enthält, ergibt sich daraus eine prozedurale Frage. Werden zunächst die im Folgezustand nicht mehr gültigen Teile des Zustands entfernt und dann die Beschreibungen der aus der Aktion resultierenden neuen positiven Fakten hinzugefügt oder umgekehrt?

Wissensrepräsentation

Beispiel: Verladestation (Fortsetzung)

- Operator (einfache Modellierung)

  operator: *move*(t, l_1, l_2)

  ;; truck t moves from location l_1 to l_2

  precond: *at*(t, l_1)

  effects: *at*$(t, l_2), \neg$*at*(t, l_1)

- Operator (komplexere Modellierung)

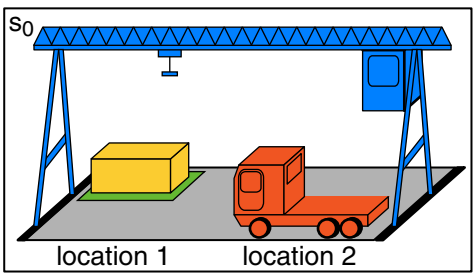
  operator : operator : *move*(t, l_1, l_2)

  ;; truck t moves from location l_1 to l_2

  precond: *at*(t, l_1), *adjacent*$(l_1, l_2), \neg$*occupied*(l_2)

  effects: *at*$(t, l_2), \neg$*at*(t, l_1), *occupied*$(l_2), \neg$*occupied*(l_1)

Anwendbare Instanzen von *move*(t, l_1, l_2)?



Wissensrepräsentation

Suchraum für Planungsprobleme in STRIPS Sprache

- Aktionen sind anwendbar, wenn der aktuelle Zustand die Vorbedingung der Aktion erfüllt:
  - die nicht-negierten Literale der Vorbedingungen in *precond* sind im Zustand wahr (enthalten),
  - die negierten Literale der Vorbedingungen in *precond* sind im Zustand wahr (nicht enthalten).

- Der Folgezustand einer anwendbaren Aktion ergibt sich aus dem aktuellen Zustand:
  - die negierten Literale der Effekte in *effects* sind im Folgezustand nicht wahr (werden aus dem aktuellen Zustand entfernt),
  - die nicht-negierten Literale der Effekte in *effects* sind im Folgezustand wahr (werden zum aktuellen Zustand hinzugefügt).

- STRIPS Assumption:
  Nur in einer Aktionen explizit (im *precond* bzw. *effects* Teil) genannte Eigenschaften sind von der Ausführung der Aktion betroffen.

  ➜ Vermeidung des Frame Problems

Bemerkungen:

- Umgekehrt bedeutet die Vermeidung des Frame Problems auch, dass die Modellierung in STRIPS Sprache nur sinnvoll ist für Aktionen, deren Wirkung in der Umgebung *lokal* ist. Kleine *effects* Teile sind wünschenswert, können aber keine umfangreichen Veränderungen (Domino-Effekte) beschreiben.
- Bezeichnen wir für einen Operator $op(x_1, \ldots, x_n)$ und eine Aktion $op(c_1, \ldots, c_n)$ mit
  - $precond^+(op(c_1, \ldots, c_n))$ die Menge der atomaren Formeln, die in der Vorbedingung von $op(c_1, \ldots, c_n)$ in nicht-negierten Literalen auftreten und
  - $precond^-(op(c_1, \ldots, c_n))$ die Menge der atomaren Formeln, die in der Vorbedingungen in negierten Literalen auftreten sowie
  - $effects^+(op(c_1, \ldots, c_n))$ die Menge der atomaren Formeln, die in den Effekten von $op(c_1, \ldots, c_n)$ in nicht-negierten Literalen auftreten und
  - $effects^-(op(c_1, \ldots, c_n))$ die Menge der atomaren Formeln, die in den Effekten in negierten Literalen auftreten.

  Dann ist eine Aktion $op(c_1, \ldots, c_n)$ im Zustand s (Menge atomarer variablenfreier Formeln) anwendbar genau dann, wenn gilt

  $$precond^+(op(c_1, \ldots, c_n)) \subseteq s \quad \text{und} \quad precond^-(op(c_1, \ldots, c_n)) \cap s = \emptyset$$

  Der Folgezustand $\tau(s, op(c_1, \ldots, c_n)$ wird beschrieben durch

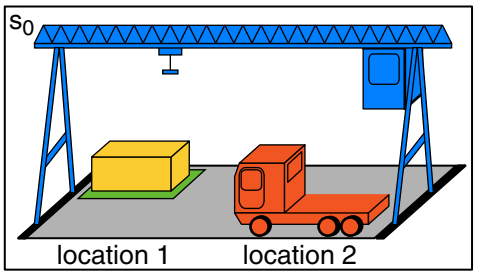
  $$\tau(s, op(c_1, \ldots, c_n) = (s \setminus effects^-(op(c_1, \ldots, c_n))) \cup effects^+(op(c_1, \ldots, c_n))$$

  (Erst löschen, dann hinzufügen!)

Wissensrepräsentation

Beispiel: Verladestation (Fortsetzung)

- Zustand
  $\{at(truck1, location2), unloaded(truck1),$
  $in(c1, pile1), top(c1, pile1), empty(crane1)\}$



- Operator

  operator: $move(t, l_1, l_2)$

  ;; truck t moves from location l_1 to l_2

  precond: $at(t, l_1)$

  effects: $at(t, l_2), \neg at(t, l_1)$

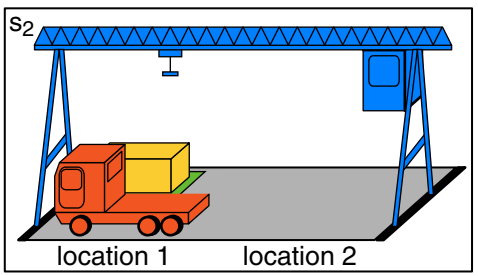
- Aktion definiert durch Substitution $[t/truck1,\ l_2/location1,\ l_1/location2]$

  action: $move(truck1, location2, location1)$

  precond: $at(truck1, location2)$

  effects: $at(truck1, location1), \neg at(truck1, location2)$

- Folgezustand
  $\{\ at(truck1, location1), unloaded(truck1),$
  $in(c1, pile1), top(c1, pile1), empty(crane1)\ \}$



Wissensrepräsentation

Suchraum für Planungsprobleme in STRIPS Sprache (Fortsetzung)

- Ein *Ziel* ist eine endliche Menge (Konjunktion) von variablenfreien Literalen.

  (Im Vergleich zum Anfangszustand ist das Ziel ein nur unvollständig spezifizierter Zustand.)

- Im aktuellen Zustand ist ein *Ziel erreicht*, wenn jedes Literal des Zieles im aktuellen Zustand wahr ist (d.h. positive Literale enthalten bzw. negative Literale nicht enthalten).

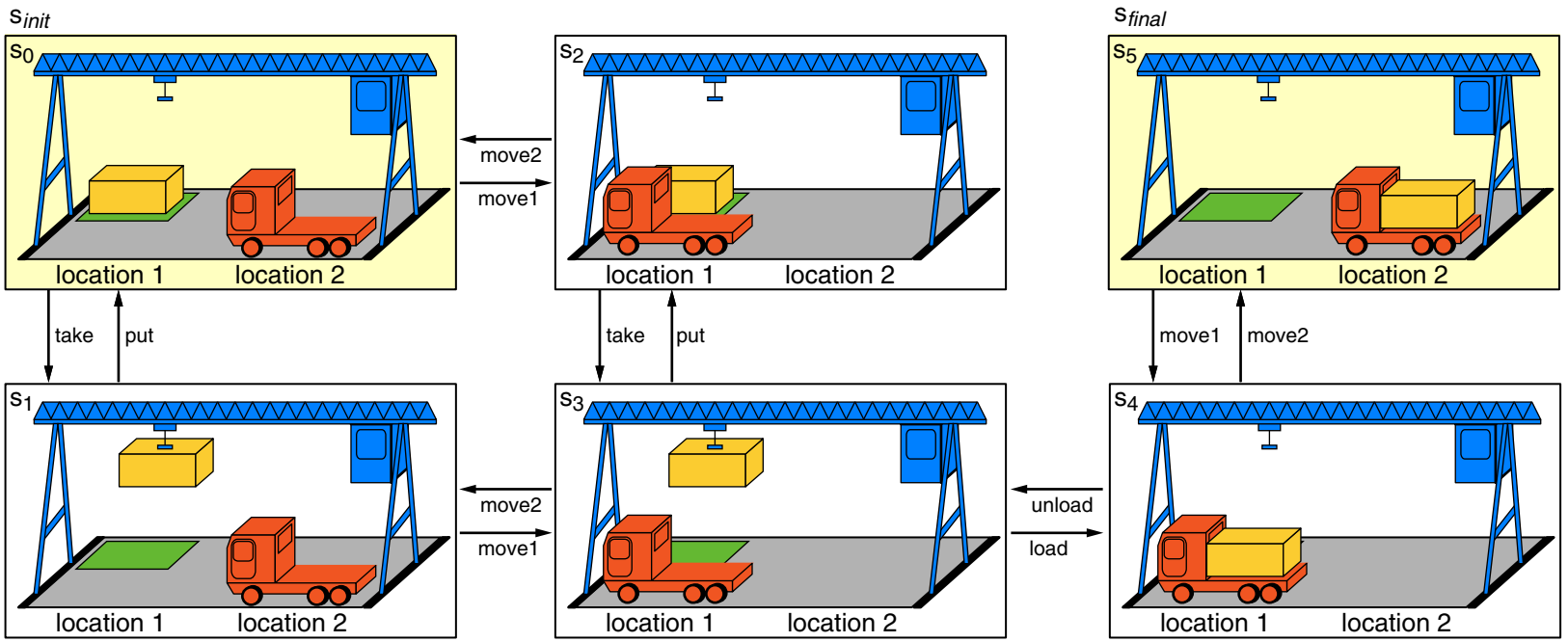
- Ein Zustand, in dem ein Ziel erreicht ist, heißt *Zielzustand*.

- Ein *Plan* ist eine Folge von Aktionen, die auf einen gegebenen Zustand angewendet werden kann und zu einem neuen Zustand führt.
  ($\rightarrow$ induktive Definition)

➜ Gesucht wird ein Plan, der von einem Anfangszustand zu einem Zielzustand führt.

Wissensrepräsentation

Beispiel: Verladestation (Fortsetzung)



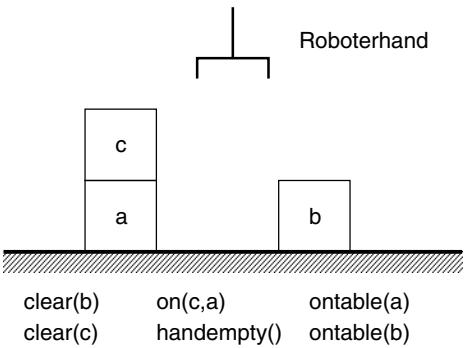
Plan:

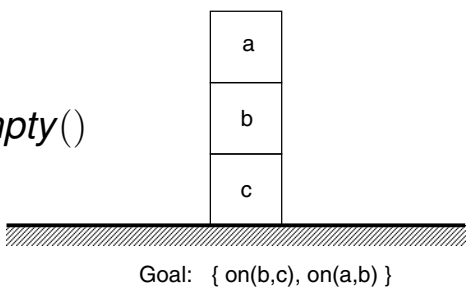
1. $move(truck1, location2, location1)$,
2. $take(crane1, c1, ground, pile1)$,
3. $load(crane1, c1, truck1)$,
4. $move(truck1, location1, location2)$

Wissensrepräsentation

Beispiel: Blocks World (nach Nilsson)

- Operatoren:
  - operator : *pickup*(x)
    - precond : *ontable*(x), *clear*(x), *handempty*$()$
    - effects : *holding*(x), $\neg$*ontable*(x), $\neg$*clear*(x), $\neg$*handempty*$()$
  - operator : *putdown*(x)
    - *precond* : *holding*(x)
    - effects : *ontable*(x), *clear*(x), *handempty*$()$, $\neg$*holding*(x)
  - operator : *stack*(x, y)
    - *precond* : *holding*(x), *clear*(y)
    - effects : *on*(x, y), *clear*(x), $\neg$*clear*(y), $\neg$*holding*(x), *handempty*$()$
  - operator : *unstack*(x, y)
    - *precond* : *on*(x, y), *clear*(x), *handempty*$()$
    - effects : *holding*(x), *clear*(y), $\neg$*on*(x, y), $\neg$*clear*(x), $\neg$*handempty*$()$





- Anfangszustand: {*clear*(b), *clear*(c), *on*(c, a), *handempty*$()$, *ontable*(a), *ontable*(b)}
- Ziel: {*on*(b, c), *on*(a, b)}
- Plan: (*unstack*(c, a), *putdown*(c), *pickup*(b), *stack*(b, c), *pickup*(a), *stack*(a, b))

Wissensrepräsentation

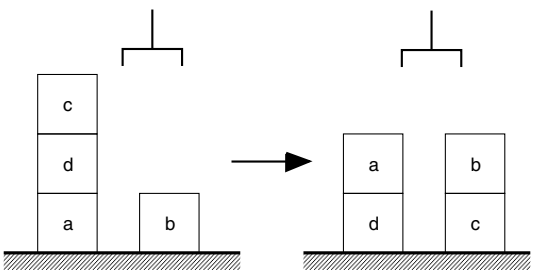
Beispiel: Blocks World (Alternative Modellierung)

- Konstanten: a, b, c, d, *floor*
- Prädikate: *on*(x, y), *clear*(x)
- Operatoren:

  operator : *move*(x, y, z)

  precond : *on*(x, y), *clear*(x), *clear*(z)

  effects : $\neg$*on*(x, y), $\neg$*clear*(z), *on*(x, z), *clear*(y), *clear*$($*floor*$)$



- Anfangszustand: {*on*(d, a), *on*(c, d), *on*$(a,$ *floor*$)$, *on*$(b,$ *floor*$)$, *clear*(b), *clear*(c), *clear*$($*floor*$)$}
- Ziel: {*on*(a, d), *on*$(d,$ *floor*$)$, *on*(b, c), *on*$(c,$ *floor*$)$}
- Notwendige Aktionen: *move*$(a,$ *floor*$, d)$, *move*$(b,$ *floor*$, c)$, *move*$(c, d,$ *floor*$)$, *move*$(d, a,$ *floor*$)$
- Plan: (*move*$(c, d,$ *floor*$)$, *move*$(d, a,$ *floor*$)$, *move*$(a,$ *floor*$, d)$, *move*$(b,$ *floor*$, c)$)

Bemerkungen:

- Ganz offensichtlich ist in dieser Modellierung nicht jede Grundinstanz eines Operators eine sinnvolle Aktion:
  - Die Substitution $[x/a, y/\textit{floor}, z/\textit{floor}]$ führt zu Konflikten im effects Teil, da das Atom $\textit{clear}(\textit{floor})$ sowohl negiert, als auch nicht-negiert auftritt.
    Die festgelegte Reihenfolge bei der Erzeugung des Folgezustandes behebt den Konflikt:

    „Erst negierte Atome im effects Teil aus dem Zustand entfernen,
    dann nicht-negierte Atome zum Zustand hinzufügen!!'
  - Die Substitution $[x/\textit{floor}, y/a, z/b]$ liefert eine unsinnige Aktion, die in keinem Zustand anwendbar ist, der eine „reale" Situation darstellt.
    Wenn man von einer „guten" Modellierung der Situatonen in der Umgebung als Zustand ausgeht und die Zustandsübergänge ebenso passend modelliert sind, können unsinnige Aktionen nicht angewendet werden.
    Alternativ können Sortenprädikate verwendet werden, z.B. $\textit{block}(x)$ im precond Teil.

- Der Kürze der Modellierung ist auch zum Opfer gefallen, dass sich die Eigenschaft $\textit{clear}(\textit{floor})$ nicht verbrauchen sollte. Dieses Atom ist einfach dem effects Teil des Operators hinzugefügt worden, aber nicht bei jeder Anwendung ist dies wirklich eine neue Eigenschaft des Folgezustandes. Wieder entsteht auch das unter dem ersten Punkt beschriebene Problem.

Bemerkungen: (Fortsetzung)

- Auch in der ersten Modellierung auf Seite 33 waren nicht alle Grundinstanzen von Operatoren sinnvoll, etwa *unstack*(a, a).
- Mögliche Alternative zur Beschreibung der Operatoren mit Sortenprädikaten und Ungleichungen in den Bedingungen und angepasstem Anfangszustand:

  operator : *move*(x, y, z)
  precond : *block*(x), *block*(z), $x \neq y, y \neq z, x \neq z$, *on*$(x, y)$, *clear*$(x)$, *clear*$(z)$
  effects : *on*(x, z), $\neg$*on*(x, y), *clear*(y), $\neg$*clear*(z)

  operator : *move*$(x, y,$ *floor*$)$
  precond : *block*(x), *block*(y), $x \neq y, y \neq$ *floor*, $x \neq$ *floor*, *on*(x, y), *clear*(x), *clear*$($*floor*$)$
  effects : *on*(x, z), $\neg$*on*(x, y), *clear*(y)

  Anfangszustand: $\{$*block*(a), *block*(b), *block*(c), *block*(d), $a = a, b = b, c = c, d = d$, *floor* $=$ *floor*, *on*(d, a), *on*(c, d), *on*$(a,$ *floor*$)$, *on*$(b,$ *floor*$)$, *clear*(c), *clear*$($*floor*$)\}$

  Die Prüfung der Ungleichungen in den precond Teilen kann man auch in das Prüfungsverfahren direkt integrieren. Dann entfallen die Gleichungen in der Zustandsbeschreibung.

Wissensrepräsentation

Klassisches Planungsproblem in STRIPS

Ein *klassisches Planungsproblem* (O, s_{init}, s_{goal}) in der STRIPS Sprache wird bestimmt durch

- eine endliche Menge von Operatoren O,
- einen Startzustand s_{init},
- und ein Ziel s_{goal}.

➜ Gesucht ist ein Plan $p = (a_1, \ldots, a_n)$, d.h. eine Folge von Aktionen, also Grundinstanzen aus O, mit $s_{init} \xrightarrow{a_1} \ldots \xrightarrow{a_n} s_{final}$ und s_{goal} ist in s_{final} erreicht.

Die *Länge* eines Planes ist die Anzahl der enthaltenen Aktionen.

Die Reihenfolge der Aktionen ist vollständig festgelegt, d.h. gesucht wird ein *linearer Plan*.

Bemerkungen:

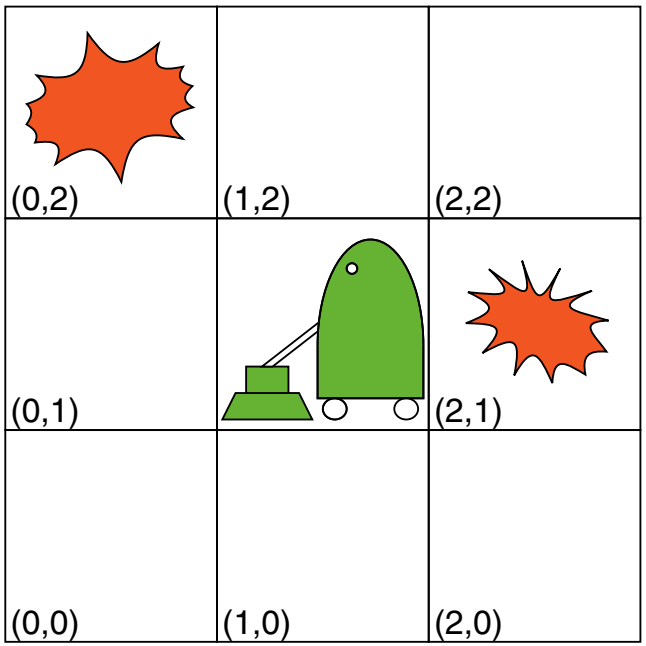
- Durch den Anfangszustand s_{init} und die Vorbedingungen *precond* und die Effekte *effects* der Operatoren in O wird implizit auch die Sprache festgelegt:
  - Nur die dort vorkommenden Prädikate sind erlaubt.
  - Nur die dort vorkommenden Konstanten sind erlaubt.
- Mit der Sprache ist auch die Menge der möglichen Grundinstanzen der Operatoren, also die Aktionen A festgelegt.
- Ausgehend vom Startzustand kann die transitive Hülle der durch die Aktionen erreichbaren Folgezustände S bestimmt werden.
- Durch Zustandsmenge und Aktionen ist das Modell der Umgebung als endliches Zustandsübergangssystem (S, A, τ) festgelegt.
- Die Menge *states*(s_{goal}) der Zielzustände ist definiert durch

  $$\textit{states}(s_{goal}) := \{s \in S \mid s_{goal} \text{ ist in } s \text{ erreicht}\}$$

  und damit eine Teilmenge der Zustandsmenge. Für sinnvolle Ziele ist *states*(s_{goal}) nicht leer.
- Die verkürzte Darstellung von klassischen Planungsproblemen ist also keine Einschränkung gegenüber der früheren Darstellung.

Wissensrepräsentation

Beispiel Vacuum World (Wooldrige, Russell/Norvig)



Sprache

- *At*(x, y) Agent ist auf Position (x, y) mit $x, y \in \{1, 2, 3\}$;
- *Dirty*(x, y) Auf Position (x, y) befindet sich Schmutz;
- *Facing*(d) Agent schaut in Richtung d mit $d \in \{$*north*, *west*, *south*, *east*$\}$;
- Aktionen $\{$*forward*, *turn*, *suck*$\}$.

Wissensrepräsentation

Beispiel Vacuum World (Wooldrige, Russell/Norvig) (Fortsetzung)

Modellierung in STRIPS Sprache

- Anfangszustand

  $s_0 = \{\mathit{At}(1,1), \mathit{Facing}(\mathit{west}), \mathit{Dirty}(0,2), \mathit{Dirty}(1,2)\}$

- Operatoren

  operator : $\mathit{turn}()$

  precond : $\mathit{Facing}(\mathit{west})$

  effects : $\mathit{Facing}(\mathit{north}), \neg\mathit{Facing}(\mathit{west})$

  ...

- Ausführung von Aktionen (Grundinstanzen von Operatoren)

  Prüfung der Anwendbarkeit:
  Gilt Grundinstanz von precond im aktuellen Zustand?

  Bildung des Folgezustands entsprechend Operatordefinition:
  Löschen der negativen Effekte, Hinzufügen der positiven Effekte

$s_1 = \tau(s_0, \mathit{turn}) = \{\mathit{At}(1,1), \mathit{Facing}(\mathit{north}), \mathit{Dirty}(0,2), \mathit{Dirty}(1,2)\}$