Wissensrepräsentation
–
Christopher Habel, Hedda Schmidtke
Sommersemester 2004

Sitzung 26: Aktionen und Instruktionen (2)

- Situationskalkül
  - Frame-Problem – Frame-Axiome
- Erweiterung: Golog:
  Programmiersprache für Agenten
  - Instruktionsbasiertes Handeln

Literatur

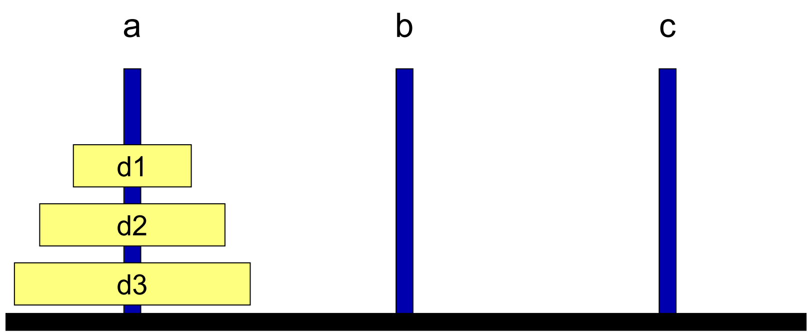
Levesque, Hector J., Raymond Reiter, Yves Lespérance, Fangzhen Lin & Richard B. Scherl (1997). GOLOG: A logic programming language for dynamic domains. Journal of Logic Programming 31. 59–84.

Brachman, Ronald J. & Hector J. Levesque (to appear). Knowledge Representation and Reasoning. Chapter 14.

Reiter, Raymond (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz (ed.) Artificial Intelligence and Mathematical Theory of Computation (pp. 359–380). Academic Press: Boston.

Aktionen für den Turm von Hanoi

Gruppen-
diskussion



Welche Relationen, Fluents, Aktionen werden benötigt?
Welche Vorbedingungen und Effekte haben die Aktionen?

Turm von Hanoi

Aktionen

- move(x, y, d) % Bewege Scheibe d von Stab x zu Stab y

Fluents

- ontop(d, x, s) % Scheibe d ist die oberste Scheibe an Stab x
- free(x, s) % Stab x ist frei
- on(d, d', s) % Scheibe d liegt auf Scheibe d'

Situationsinvariante Relationen

- d < d' % Scheibe d ist kleiner als Scheibe d'

Vorbedingungen

- $Poss(move(x, y, d), s) \Leftrightarrow ontop(d, x, s) \wedge$
  $\neg\exists d' [ontop(d', y, s) \wedge d' < d]$

Turm von Hanoi

Hintergrundwissen
- Stäbe: a, b, c
- Scheiben: d1, d2, d3
- Größenverhältnisse: d1 < d2, d2 < d3, d1 < d3

Startsituation
- ontop(d1, a, s0)
- free(b, s0)
- free(c, s0)
- on(d1, d2, s0)
- on(d2, d3, s0)

Revision: Aufzug

Primitive actions

turnoff(n)	% Turn off call button n.
open	% Open elevator door.
close	% Close elevator door.
up(n)	% Move elevator up to floor n.
down(n)	% Move elevator down to floor n.

Fluents

on(n, s)	% unanswered call on floor n
currentFloor(n, s)	% current floor has number n
+ d_closed(s)	% the elevator door is closed

Revision: Vorbedingungen

Preconditions for Primitive Actions

$\mathcal{Pre}_{EL}$

∀s, n [poss(turnoff(n),s) ⇔ on(n, s) ∧
currentFloor(n, s) ∧ ¬d_closed(s)]
% Anforderung befriedigt wenn mit offener Tür auf Etage

∀s [poss(open,s) ⇔ d_closed(s)] % nur geschl. Tür öffnen

∀s [poss(close,s) ⇔ ¬d_closed(s)] % nur offene Tür schließen

∀s, n, m [poss(up(n), s) ⇔ currentFloor(m, s) ∧ m < n ∧
d_closed(s)] %Fahrt nur mit geschlossener Tür

∀s, n, m [poss(down(n), s) ⇔ currentFloor(m, s) ∧ m > n ∧
d_closed(s)] %Fahrt nur mit geschlossener Tür

Revision: Effekte

Effect Axioms for Primitive Fluents

$\mathcal{Eff}_{EL}$

- **Positive Effekte**
- ∀s [d_closed(do(close, s))] % Tür zu
- ∀s, m [currentFloor(m, do(up(m), s))] % Position
- ∀s, m [currentFloor(m, do(down(m), s))] % Position
- **Negative Effekte**
- ∀s, m [¬on(m, do(turnoff(m), s))] % Anforderung aus
- ∀s [¬d_closed(do(open, s))] % Tür auf
- ∀s,n,m [n≠m ⇒ ¬currentFloor(m,do(up(n),s))] % Position
- ∀s,n,m [n≠m ⇒ ¬currentFloor(m,do(down(n),s))] % Position

Mögliche Aktionen

Initiale Situation s0

Call buttons: 3 and 5. Elevator is at floor 4. Door is open.

$\mathcal{Init}_{EL}$: on(3, s_0), on(5, s_0), currentFloor(4, s_0)

Mögliche Aktionen in

•s_0	•do(close, s_0)	•do(up(5), do(close, s_0))	•do(open, do(up(5), do(close, s_0)))
•close	•open •up(5) •down(3) •down(2) •down(1)	•open •down(4) •down(3) •down(2) •down(1)	•close •turnoff(5)

?

$\mathcal{Pre}_{EL} \cup \mathcal{Eff}_{EL} \cup \mathcal{Init}_{EL} \models$ poss(close, s_0) ?

∀s [poss(close, s) ⇔ ¬d_closed(s)] $\in \mathcal{Pre}_{EL}$

Initiale Situation

- Call buttons: 3 and 5. Elevator is at floor 4. Door is open.

  on(3, s_0), on(5, s_0), currentFloor(4, s_0) $\in \mathcal{Init}_{EL}$
- Closed World Assumption (?)

Vollständige Spezifikation der initialen Situation

$\mathcal{Init}_{EL}$

∀ m [on(m, s_0) ⇔ m = 3 ∨ m = 5]

∀ m [currentFloor(m, s_0) ⇔ m = 4]

¬d_closed(s_0)

$\mathcal{Pre}_{EL} \cup \mathcal{Eff}_{EL} \cup \mathcal{Init}_{EL} \models$ poss(open, do(close, s_0)) ?

∀s [poss(open, s) ⇔ d_closed(s)] $\in \mathcal{Pre}_{EL}$

Positives Effekt-Axiom

- ∀s [d_closed(do(close, s))] $\in \mathcal{Eff}_{EL}$

- **Also gilt sogar**
  - $\mathcal{Pre}_{EL} \cup \mathcal{Eff}_{EL} \models$ ∀s [poss(open, do(close, s))]
  - (Was auch immer ist, wenn man gerade die Tür geöffnet hat, dann kann man sie auch wieder schließen)

$\mathcal{Pre}_{EL} \cup \mathcal{Eff}_{EL} \cup \mathcal{Init}_{EL} \models$ poss(up(5), do(close, s_0)) ?

∀s, n, m [poss(up(n), s) ⇔ currentFloor(m, s) ∧ m < n ∧ d_closed(s)] $\in \mathcal{Pre}_{EL}$

- Gilt $\mathcal{Pre}_{EL} \cup \mathcal{Eff}_{EL} \cup \mathcal{Init}_{EL} \models$ currentFloor(m, do(close, s_0)) für irgendein m?

- **Initiale Situation**
  - ∀ m [currentFloor(m, s_0) ⇔ m = 4] $\in \mathcal{Init}_{EL}$
- **Effekt-Axiome zu** currentFloor: $\mathcal{Eff}_{EL}$
  - ∀s, m [currentFloor(m, do(up(m), s))]
  - ∀s, n, m [n ≠ m ⇒ ¬currentFloor(m, do(up(n), s))]
  - ∀s, m [currentFloor(m, do(down(m), s))]
  - ∀s, n, m [n ≠ m ⇒ ¬currentFloor(m, do(down(n), s))]
  - Das Schließen der Tür hat keinen Effekt auf die Position des Aufzugs

Frame-Problem

McCarthy & Hayes (1969)

Effekte von Handlungen

- sind (in der Regel) beschränkt
  - Das Fallenlassen des Stiftes verändert nicht seine Farbe.
  - Das Schließen der Tür verändert nicht die Position des Aufzugs.
- und die Beschränkungen sind uns (weitgehend) bekannt.

Frame-Axiome

- machen Aussagen über Invarianz von Fluents
- (Kein-)Effekt-Axiome
- Sind im Normalfall nicht aus Effekt-Axiomen folgerbar.

Im Aufzug-Beispiel

- Das Öffnen und Schließen der Tür und das Ausschalten der Anforderung verändert die Position des Fahrstuhls nicht.

$\mathcal{Fr}_{EL}$

$\forall$s, m [currentFloor(m,do(turnoff(n),s)) $\Leftrightarrow$ currentFloor(m,s)]

$\forall$s, m [currentFloor(m,do(open,s)) $\Leftrightarrow$ currentFloor(m,s)]

$\forall$s, m [currentFloor(m,do(close,s)) $\Leftrightarrow$ currentFloor(m,s)]

$\mathcal{Pre}_{EL} \cup \mathcal{Eff}_{EL} \cup \mathcal{Fr}_{EL} \cup \mathcal{Init}_{EL} \models$ **poss(up(5), do(close, s_0))**

$\forall$s, n, m [poss(up(n), s) $\Leftrightarrow$ currentFloor(m, s) $\wedge$ m < n $\wedge$ d_closed(s)] $\in \mathcal{Pre}_{EL}$

- Gilt $\mathcal{Pre}_{EL} \cup \mathcal{Eff}_{EL} \cup \mathcal{Fr}_{EL} \cup \mathcal{Init}_{EL} \models$ currentFloor(m, do(close, s_0)) für irgendein m?

- **Initiale Situation**
  - $\forall$m [currentFloor(m, s_0) $\Leftrightarrow$ m = 4] $\in \mathcal{Init}_{EL}$
- **Frame Axiom**

  $\forall$s,m[currentFloor(m,do(close,s))$\Leftrightarrow$currentFloor(m,s)] $\in \mathcal{Fr}_{EL}$

Also gilt

$\mathcal{Fr}_{EL} \cup \mathcal{Init}_{EL} \models$ currentFloor(4, do(close, s_0))

$\mathcal{Pre}_{EL} \cup \mathcal{Eff}_{EL} \cup \mathcal{Fr}_{EL} \cup \mathcal{Init}_{EL} \models$ poss(up(5), do(close, s_0))

Weitere Frame-Axiome

$\mathcal{Eff}_{EL}$

$\forall$s, m [$\neg$on(m, do(turnoff(m), s))]

$\forall$s [d_closed(do(close, s))]

$\forall$s [$\neg$d_closed(do(open, s))]

$\mathcal{Fr}_{EL}$

$\forall$s, m [on(m,do(open,s)) $\Leftrightarrow$ on(m,s)]

$\forall$s, m [on(m,do(close,s)) $\Leftrightarrow$ on(m,s)]

$\forall$s, m, n [on(m,do(up(n),s)) $\Leftrightarrow$ on(m,s)]

$\forall$s, m, n [on(m,do(down(n),s)) $\Leftrightarrow$ on(m,s)]

$\forall$s, m [d_closed(do(turnoff(m),s)) $\Leftrightarrow$ d_closed(s)]

$\forall$s, m [d_closed(do(up(m),s)) $\Leftrightarrow$ d_closed(s)]

$\forall$s, m [d_closed(do(down(m),s)) $\Leftrightarrow$ d_closed(s)]

Frame-Axiome

Mögliches Problem

- Kein-Effekt ist der Normalfall.
- Bei n primitiven Aktionen und m primitiven Fluents ergeben sich $\mathcal{O}$(m*n) Frame-Axiome
- (Wie) lassen sich die Frame-Axiome automatisch bestimmen?

Closed World Assumption für Fluents und Aktionen

- Wenn ein Fluent durch eine Aktion nicht verändert wird, dann hat er nach der Durchführung denselben Wert wie vorher.
- Bei vollständigem Wissen über die Effekte aller primitiven Aktionen ist die Bestimmung der Frame-Axiome möglich.

Bestimmung der Frame-Axiome

Gegeben

- ein Fluent Φ
- Vollständige Spezifikation der positiven und negativen Effekte aller Handlungen bzgl. Φ

Ziel

- Generierung der Frame-Axiome für Φ
- Darstellung in kompakter Form

Lösung

- Zusammenfassung der Effekt-Axiome
- Generierung der Erklärungsabschlussaxiomen (explanation closure)
- Bestimmung von Nachfolgezustandsaxiomen (successor state axioms)

Bestimmung der Frame-Axiome: Vorbereitung

- bringe die positiven und negativen Effekt-Axiome zu Φ in folgende Normalform,
  $\forall x, s\ [\Psi_i(x, s) \Rightarrow \Phi(x, do(a_i, s))]$
  $\forall x, s\ [\Theta_j(x, s) \Rightarrow \neg\Phi(x, do(b_j, s))]$
  s eine Situationsvariable, x ein Variablen-Vektor, a_i, b_i Namen von Aktionen, $\Psi_i(x, s)$, $\Theta_j(x, s)$ Formeln mit x und s als freie Variablen
- Schreibe die Normalformen um (mit a, b neue Variablen)
  $\forall x, s, a\ [\Psi_i(x, s) \wedge a = a_i \Rightarrow \Phi(x, do(a, s))]$
  $\forall x, s, b\ [\Theta_j(x, s) \wedge b = b_j \Rightarrow \neg\Phi(x, do(b, s))]$
- Fasse positive bzw. negative Effekte zusammen zu
  $\forall x, s, a\ [\Pi_\Phi(x, a, s) \Rightarrow \Phi(x, do(a, s))]$
  $\forall x, s, b\ [N_\Phi(x, b, s) \Rightarrow \neg\Phi(x, do(b, s))]$

Beispiel: on(x, s)

- Effekt-Axiom: $\forall s, x\ [\neg on(x, do(turnoff(x), s))]$
- bringe das negative Effekt-Axiom in Normalform,
  $\forall x, s\ [\Theta_j(x, s) \Rightarrow \neg\Phi(x, do(b_j, s))]$
  $\forall x, s [\top \Rightarrow \neg on(x, do(turnoff(x), s))]$
- Schreibe die Normalformen um
  $\forall x, s, b\ [\Theta_j(x, s) \wedge b = b_j \Rightarrow \neg\Phi(x, do(b, s))]$
  $\forall x, s, b\ [b = turnoff(x) \Rightarrow \neg on(x, do(b, s))]$
- Fasse positive bzw. negative Effekte zusammen zu
  $\forall x, s, a\ [\Pi_\Phi(x, a, s) \Rightarrow \Phi(x, do(a, s))]$
  $\forall x, s, a\ [\perp \Rightarrow on(x, do(a, s))]$
  $\forall x, s, b\ [N_\Phi(x, b, s) \Rightarrow \neg\Phi(x, do(b, s))]$
  $\forall x, s, b\ [b = turnoff(x) \Rightarrow \neg on(x, do(b, s))]$

Beispiel: d_closed(s)

- Effekt-Axiome
  $\forall s\ [d\_closed(do(close, s))]$
  $\forall s\ [\neg d\_closed(do(open, s))]$
- bringe die Effekt-Axiom in Normalform
  $\forall x, s\ [\Psi_i(x, s) \Rightarrow \Phi(x, do(a_i, s))]$
  $\forall s\ [\top \Rightarrow d\_closed(do(close, s))]$
  $\forall x, s\ [\Theta_j(x, s) \Rightarrow \neg\Phi(x, do(b_j, s))]$
  $\forall x, s\ [\top \Rightarrow \neg d\_closed(do(open, s))]$
- Schreibe die Normalformen um
  $\forall x, s, a\ [\Psi_i(x, s) \wedge a = a_i \Rightarrow \Phi(x, do(a, s))]$
  $\forall s, a\ [a = close \Rightarrow d\_closed(do(a, s))]$
  $\forall x, s, b\ [\Theta_j(x, s) \wedge b = b_j \Rightarrow \neg\Phi(x, do(b, s))]$
  $\forall s, b\ [b = open \Rightarrow \neg d\_closed(do(b, s))]$

Beispiel: currentFloor(x, s)

- Effekt-Axiome: $\forall s, x\ [currentFloor(x, do(up(x), s))]$
  $\forall s, x\ [currentFloor(x, do(down(x), s))]$
  $\forall s, n, x\ [n \neq x \Rightarrow \neg currentFloor(x, do(up(n), s))]$
  $\forall s, n, x\ [n \neq x \Rightarrow \neg currentFloor(x, do(down(n), s))]$
- Normalform
  $\forall x, s\ [\top \Rightarrow currentFloor(x, do(up(x), s))]$
  $\forall x, s\ [\top \Rightarrow currentFloor(x, do(down(x), s))]$
  $\forall x, n, s\ [n \neq x \Rightarrow \neg currentFloor(x, do(up(n), s))]$
  $\forall x, n, s\ [n \neq x \Rightarrow \neg currentFloor(x, do(down(n), s))]$
- Schreibe die Normalformen um
  $\forall x, s, a\ [a = up(x) \Rightarrow currentFloor(x, do(a, s))]$
  $\forall x, s, a\ [a = down(x) \Rightarrow currentFloor(x, do(a, s))]$
  $\forall x, s, b\ [\exists n\ [n \neq x \wedge b = up(n)] \Rightarrow \neg currentFloor(x,do(b,s))]$
  $\forall x, s, b\ [\exists n\ [n{\neq}x \wedge b{=}down(n)] \Rightarrow \neg currentFloor(x,do(b,s))]$

Beispiel: currentFloor(x, s) (Forts.)

- Fasse positive bzw. negative Effekte zusammen zu
  $\forall x, s, a\ [\Pi_\Phi(x, a, s) \Rightarrow \Phi(x, do(a, s))]$
  $\forall x, s, a\ [a = up(x) \vee a = down(x)$
  $\quad \Rightarrow currentFloor(x,do(a,s))]$
  $\forall x, s, b\ [N_\Phi(x, b, s) \Rightarrow \neg\Phi(x, do(b, s))]$
  $\forall x, s, b\ [\exists n\ [(n \neq x \wedge b = up(n)) \vee (n \neq x \wedge b = down(n))]$
  $\quad \Rightarrow \neg currentFloor(x, do(b, s))]$

Bestimmung der Frame-Axiome: Erklärungsabschluss

zusammengefasste Effekt-Axiome

$\forall x, s, a\ [\Pi_\Phi(x, a, s) \Rightarrow \Phi(x, do(a, s))]$
$\forall x, s, b\ [N_\Phi(x, b, s) \Rightarrow \neg\Phi(x, do(b, s))]$

Annahme

- Alle Effekte sind explizit kodiert.

Erklärungsabschlussaxiome (explanation closure) $\mathcal{Fr}$

- Erkläre jeden Wechsel durch bekannte Effekte
  $\forall x, s, a\ [\neg\Phi(x, s) \wedge \Phi(x, do(a, s)) \Rightarrow \Pi_\Phi(x, a, s)]$
  $\forall x, s, b\ [\Phi(x, s) \wedge \neg\Phi(x, do(b, s)) \Rightarrow N_\Phi(x, b, s)]$
- in Frame-Axiom-Form: Fehlen Voraussetzungen für Effekte, dann bleibt alles wie gehabt
  $\forall x, s, a\ [\neg\Phi(x, s) \wedge \neg\Pi_\Phi(x, a, s) \Rightarrow \neg\Phi(x, do(a, s))]$
  $\forall x, s, b\ [\Phi(x, s) \wedge \neg N_\Phi(x, b, s) \Rightarrow \Phi(x, do(b, s))]$

Beispiel: on(x, s)

zusammengefasste Effekt-Axiome

$\forall x, s, a\ [\bot \Rightarrow on(x, do(a, s))]$

$\forall x, s, b\ [b = turnoff(x) \Rightarrow \neg on(x, do(b, s))]$

Erklärungsabschlussaxiome $\mathcal{Fr}_{EL}$

- Erkläre jeden Wechsel durch bekannte Effekte
  $\forall x, s, a\ [\neg on(x, s) \wedge on(x, do(a, s)) \Rightarrow \bot]$
  $\forall x, s, b\ [on(x, s) \wedge \neg on(x, do(b, s)) \Rightarrow b = turnoff(x)]$
- in Frame-Axiom-Form: Fehlen Voraussetzungen für Effekte, dann bleibt alles wie gehabt
  $\forall x, s, a\ [\neg on(x, s) \Rightarrow \neg on(x, do(a, s))]$
  $\forall x, s, b\ [on(x, s) \wedge b \neq turnoff(x) \Rightarrow on(x, do(b, s))]$

Beispiel: d_closed(s)

Bestimmung der Frame-Axiome: Erklärungsabschluss

- zusammengefasste Effekt-Axiome
  $\forall x, s, a\ [a = close \Rightarrow d\_closed(do(a, s))]$
  $\forall x, s, b\ [b = open \Rightarrow \neg d\_closed(do(b, s))]$

Erklärungsabschlussaxiome $\mathcal{Fr}_{EL}$

- Erkläre jeden Wechsel durch bekannte Effekte
  $\forall x, s, a\ [\neg d\_closed(s) \wedge d\_closed(do(a, s)) \Rightarrow a = close]$
  $\forall x, s, b\ [d\_closed(s) \wedge \neg d\_closed(do(b, s)) \Rightarrow b = open]$
- in Frame-Axiom-Form: Fehlen Voraussetzungen für Effekte, dann bleibt alles wie gehabt
  $\forall x, s, a\ [\neg d\_closed(s) \wedge a \neq close \Rightarrow \neg d\_closed(do(a, s))]$
  $\forall x, s, b\ [d\_closed(s) \wedge b \neq open \Rightarrow d\_closed(do(b, s))]$

Beispiel: currentFloor(x, s) (Forts.)

Bestimmung der Frame-Axiome: Erklärungsabschluss

- zusammengefasste Effekt-Axiome
  $\forall x, s, a\ [a = up(x) \vee a = down(x) \Rightarrow currentFloor(x,do(a,s))]$
  $\forall x,s,b [\exists n\ [n \neq x \wedge (b=up(n) \vee b=down(n))] \Rightarrow \neg cFl(x,do(b,s))]$

Erklärungsabschlussaxiome $\mathcal{Fr}_{EL}$

$\forall x, s, a\ [\neg currentFloor(x, s) \wedge currentFloor(x, do(a, s)) \Rightarrow$
$a = up(x) \vee a = down(x)]$

$\forall x, s, b\ [currentFloor(x, s) \wedge \neg currentFloor(x, do(b, s)) \Rightarrow$
$\exists n\ [n \neq x \wedge (b = up(n) \vee b = down(n))]]$

- in Frame-Axiom-Form
  $\forall x,s,a\ [\neg cFl(x,s) \wedge a \neq up(x) \wedge a \neq down(x)) \Rightarrow \neg cFl(x,do(a,s))]$
  $\forall x,s,b\ [cFl(x,s) \wedge \forall n\ [(n=x \vee (b \neq up(n) \wedge b \neq down(n)))] \Rightarrow$
  $cFl(x,do(b,s))]$

Axiome des Nachfolgezustands

zusammengefasste Effekt-Axiome

$\forall s, x, a\ [\Pi_\Phi(x, a, s) \Rightarrow \Phi(x, do(a, s))]$

$\forall s, x, b\ [N_\Phi(x, b, s) \Rightarrow \neg\Phi(x, do(b, s))]$

Annahme

- Alle Effekte sind explizit kodiert.
- Integrität: $\mathcal{KB} \models \forall s, x, a\ [\neg(\Pi_\Phi(x, a, s) \wedge N_\Phi(x, a, s))]$

Erklärungsabschlussaxiome (explanation closure)

$\forall s, x, a\ [\neg\Phi(x, s) \wedge \Phi(x, do(a, s)) \Rightarrow \Pi_\Phi(x, a, s)]$

$\forall s, x, b\ [\Phi(x, s) \wedge \neg\Phi(x, do(b, s)) \Rightarrow N_\Phi(x, b, s)]$

Axiom des Nachfolgezustands (successor state axiom)

- Fasst Effekt-Axiome und Erklärungsabschluss zusammen
  $\forall s,x,a\ [\Phi(x,do(a,s)) \Leftrightarrow \Pi_\Phi(x,a,s) \vee (\Phi(x,s) \wedge \neg N_\Phi(x,a,s))]$

Axiome des Nachfolgezustands

Axiom des Nachfolgezustands (successor state axiom)

$\forall s,x,a\ [\Phi(x,do(a,s)) \Leftrightarrow \Pi_\Phi(x,a,s) \vee (\Phi(x,s) \wedge \neg N_\Phi(x,a,s))]$

Beispiel: on(x, s)

$\forall s,x,a\ [on(x,do(a,s)) \Leftrightarrow (on(x,s) \wedge a \neq turnoff(x))]$

Beispiel: d_closed(s)

$\forall s,a\ [d\_closed(do(a,s)) \Leftrightarrow a=close$
$\vee (d\_closed(x,s) \wedge a \neq open)]$

Beispiel: currentFloor(x, s)

$\forall s,x,a\ [currentFloor(x,do(a,s)) \Leftrightarrow a = up(x) \vee a = down(x) \vee$
$(currentFloor(x,s) \wedge \neg \exists n\ [n \neq x \wedge (a=up(n) \vee a=down(n))])]$

Axiome im Situationskalkül

Vorbedingungsaxiome $\mathcal{Pre}$

- Für jede Aktion genau eine Äquivalenz
  $\forall s,a\ [Poss(a, s) \Leftrightarrow \Psi(s)]$

Axiome des Nachfolgezustands $\mathcal{SSt}$

- Zusammenfassung von
  Effekt-Axiomen $\mathcal{Eff}$
  Frame-Axiomen $\mathcal{Fr}$
- Für jeden primitiven Fluent genau eine Äquivalenz
  $\forall s,x,a\ [\Phi(x,do(a,s)) \Leftrightarrow \Pi_\Phi(x,a,s) \vee (\Phi(x,s) \wedge \neg N_\Phi(x,a,s))]$

Spezifikation des Initialzustands $\mathcal{Init}$

- s_0 als einziger Situationsausdruck

Lösung des Frame-Problems im Situationskalkül

Vollständige Spezifikation von (primitiven) Aktionen

- Vorbedingungen
- Effekte
  - Integrität (keine inkonsistenten Effekte)
  - Determiniertheit (keine disjunktiven Effekte)

Spezifikation der (primitiven) Fluents

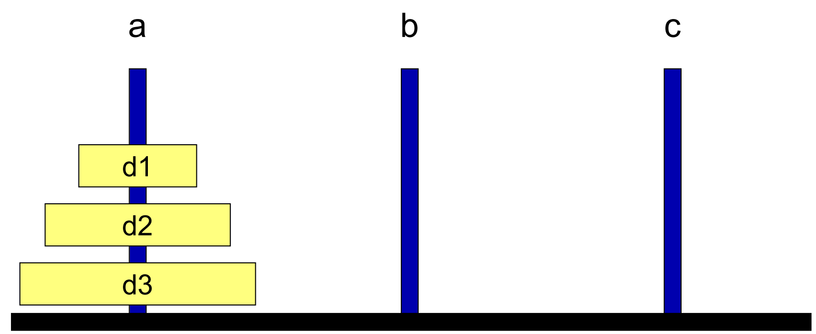
- Nachfolgezustandsaxiome generiert aus Effektspezifikationen der Aktionen (und Vollständigkeitsannahme)
- Keine Abhängigkeiten

(Annahme der eindeutigen Benennung von Aktionen)

- Brachman und Levesque nennen sie als wesentlich, ohne den Grund anzugeben.

Aktionen für den Turm von Hanoi

Gruppen-diskussion



Welche Axiome des Nachfolgezustands gelten für die Fluents?

Planung im Situationskalkül

Gegeben

- Zielspezifikation: Eine Formel G(s) mit s als einziger freien Variable
- Startsituation: eine Situation s_0 (mit vollständiger Spezifikation)

Gesucht

- Eine Folge $\langle a_1, \ldots, a_n \rangle$ von primitiven Aktionen, so dass
- $\mathcal{KB} \models G(do(\langle a_1, \ldots, a_n \rangle, s_0)) \wedge Legal(\langle a_1, \ldots, a_n \rangle, s_0))$

Verfahren

- KI-Suchverfahren (Bei Bewertungsmöglichkeit: A*)
- (Resolutions-)Beweis mit Antwortextraktion
- Goal-Regression (Reiter 1991)

Goal-Regression

Ziel

- Bestimme eine Folge $\langle a_1, \ldots, a_n \rangle$ von primitiven Aktionen, so dass
- $\mathcal{KB} \models G(do(\langle a_1, \ldots, a_n \rangle, s_0)) \wedge Legal(\langle a_1, \ldots, a_n \rangle, s_0))$

Vorgehen

- Systematische Ersetzung von Teil-Ausdrücken der Form do(a, s) in den auftretenden Fluents
  - unter Verwendung der Axiome des Nachfolgezustands und der Vorbedingungsaxiome,
  - so dass $G^*(s_0)$ nur s_0 als Situationsausdruck enthält
  - und obige Folgerungsbeziehung genau dann besteht, wenn
  - $\mathcal{KB} \setminus (\mathcal{Pre} \cup \mathcal{SSt}) \models G^*(s_0)$
- Erlaubt auch die Bestimmung von Vorbedingungen von Aktionsfolgen

Aufzugs-Planung im Situationskalkül

Gegeben

- Ziel: Alle Anforderungen erfüllt, Parken im Erdgeschoss mit offener Tür.
- $\neg\exists n\, [on(n, s) \wedge currentFloor(0, s) \wedge \neg d\_closed(s)]$
- Startsituation s_0

Gesucht

- Eine Folge $\langle a_1, \ldots, a_n \rangle$ von primitiven Aktionen, so dass
- $\mathcal{KB} \models G(do(\langle a_1, \ldots, a_n \rangle, s_0)) \wedge Legal(\langle a_1, \ldots, a_n \rangle, s_0))$

Lösung

⟨close, down(3), open, turnoff(3), close, up(5), open, turnoff(5), close, down(0), open⟩

$s = do(open, do(down(0), do(close, do(turnoff(5), do(open, do(up(5), do(close, do(turnoff(3), do(open, do(down(3), do(close, s_0)))))))))))$

Aufzugsbeispiel: Mögliche Aktionen

		•cF	•on	•d_c	•Poss(a)
•s0		•4	•3, 5	•f	•close
•s1	•do(close, s0)	•4	•3, 5	•t	•open, down(0)…(3), up(5)
•s2	•do(down(3), s1)	•3	•3, 5	•t	•open, down(0)…(2), up(4)…(5)
•s3	•do(open, s2)	•3	•3, 5	•f	•turnoff, close
•s4	•do(turnoff(3), s3)	•3	•5	•f	•close
•s5	•do(close, s4)	•3	•5	•t	•open, down(0)…(2), up(4)…(5)
•s6	•do(up(5), s5)	•5	•5	•t	•open, down(0)…(4)
•s7	•do(open, s6)	•5	•5	•f	•turnoff, close
•s8	•do(turnoff(5), s7)	•5		•f	•close
•s9	•do(close, s8)	•5		•t	•open, down(0)…(4)
•s10	•do(down(0), s9)	•0		•t	•open, up(1)…(5)
•s11	•do(open, s10)	•0		•f	•close

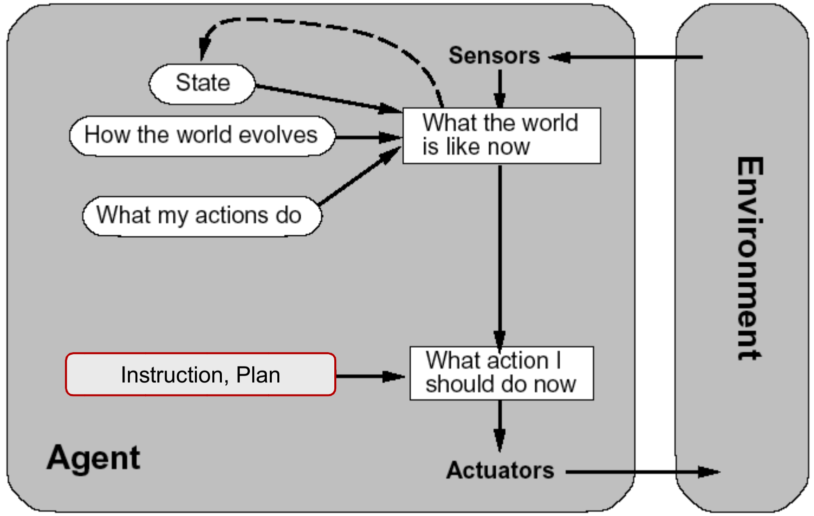
Instruktionen für einen Agenten

Als Ziel
- Spezifikation des durch den Agenten zu erreichenden Zustandes

Als Aktionssequenz
- Resultat der Planung
- durch den Agenten in dieser Form auszuführen

Als komplexe Aktion
- mit Kontrollstruktur
  - z.B. bedingten Aktionen
- Agenten-Programm
  - GOLOG

Model-based reflex agent (with instructions)



Russell & Norvig - ch.2, f.20

Komplexe Aktionen

Sequenz
- Fahre in die dritte Etage und öffne dann die Tür.

Bedingte Aktion
- Wenn keine Anforderung besteht, warte im Erdgeschoss mit offener Tür.

Iteration
- Solange ein Anforderung besteht, bediene Anforderungen.

Auswahl
- Wähle eine bestehende Anforderung zur Bedienung aus.

➔ Kontrollstrukturen in Programmiersprachen
- Unterschied: Primitive Aktionen

Komplexe Aktionen in GOLOG

Erweiterung des Situationskalküls
- Es sei
  - $\mathcal{PrA}$ eine Menge von Symbolen für primitive Aktionen
  - $\mathcal{Fl}$ eine Menge von Symbolen für Fluents
- Die Menge der GOLOG-Programme $Golog(\mathcal{PrA}, \mathcal{Fl})$ ist dann die kleinste Menge, für die gilt
  - $\mathcal{PrA} \subseteq Golog(\mathcal{PrA}, \mathcal{Fl})$
  - Wenn A, B $\in Golog(\mathcal{PrA}, \mathcal{Fl})$, $\Phi \in \mathcal{Fl}$ und x eine Variable, dann sind auch [A ; B], [A | B], if(Φ, A, B), while(Φ, A), ?(Φ), [πx. A] $\in Golog(\mathcal{PrA}, \mathcal{Fl})$

Fluents und komplexe Aktionen

Fluents für Tests

- in den Kontrollstrukturen if, while, ? werde Tests vorgenommen, deren Resultat situationsabhängig sein kann
- Fluents erlauben diesen Situationsbezug
- Damit im 'Programm' implizit gelassen werden kann, auf welche Situation der Fluent angewendet wird:
  - Φ steht für ein Fluent mit unterdrückten Situationsargument (d_open, on(3), currentFloor(4))
  - Φ(s) steht für den entsprechenden Fluent mit 'restauriertem' Situationsargument s (d_open(s), on(3, s), currentFloor(4, s))

Ausführung komplexer Aktionen

- ist eine Sequenz primitiver Aktionen, angewendet auf s0

Vorbedingungen

- ergeben sich aus den Vorbedingungen der eingebetteten primitiven Aktionen

Zielsituationen

- sind nicht eindeutig (mehrere Sequenzen primitiver Aktionen können einer komplexen Aktion entsprechen)

Do(A, s, s')

- Formel im Situationskalkül
- Intendierte Bedeutung: die Ausführung von A kann von s zu s' führen
- s' ergibt sich aufgrund einer Sequenz von primitiven Aktionen, angewendet auf s

Bedingungen für Do(A, s, s')

- 'rekursive Definition'

Primitive Aktionen

Do(a, s, s') ⇔ Poss(a, s) ∧ s' = do(a, s)

Sequenzen

Do([A;B], s, s') ⇔ ∃s'' [Do(A, s, s'') ∧ Do(B, s'', s')]

Wahl, nichtdeterministische Verzweigung

Do([A | B], s, s') ⇔ Do(A, s, s') ∨ Do(B, s, s')

Wahl eines Wertes, nichtdeterministisch

(x kommt in A frei vor)

Do([πx. A], s, s') ⇔ ∃x [Do(A, s, s')]

Bedingungen für Do(A, s, s')

Tests

Do(?(Φ), s, s') ⇔ (Φ(s) ∧ s = s')

Bedingte Aktionen

Do(if(Φ, A, B), s, s') ⇔ (Φ(s) ∧ Do(A, s, s')) ∨
(¬Φ(s) ∧ Do(B, s, s'))

Iteration, While-Schleife

Do(while(Φ, A), s, s') ⇔ (¬Φ(s) ∧ s = s') ∨
(Φ(s) ∧ Do([A;while(Φ, A)], s, s'))

- hier ist die allgemeine Terminationsproblematik zu beachten

GOLOG-Programme

Primitive Aktionen und Fluents
- sind anwendungsbezogen zu spezifizieren
- durch Vorbedingungen und Nachfolgezustandsaxiome

Die Programm-Semantik
- (= was das Ausführen der Programme bewirkt)
- ist von diesen Spezifikationen abhängig

Zwei Aspekte der Ausführung von A beginnend bei s_0
- gibt es (überhaupt) eine Sequenz von primitiven Aktionen $\langle a_1, \ldots, a_n\rangle$, so dass
  $\mathcal{KB} \models Do(A, s_0, do(\langle a_1, \ldots, a_n\rangle\ s_0))$
- Führe die Sequenz $\langle a_1, \ldots, a_n\rangle$ durch einen Roboter oder Simulator aus

Aufzug-Kontrolle

Definitionen komplexer Aktionen
goFloor(n) $=_{def}$?(currentFloor(n)) |
((?(d_closed) | close) ; (up(n) | down(n)))

serve(n) $=_{def}$ goFloor(n) ; (?(- d_closed) | open) ;
turnoff(n)

serveAfloor $=_{def}$ [πn. ?(on(n)) ; serve(n)]

park $=_{def}$ goFloor(0) ; (?(- d_closed) | open)

control $=_{def}$ while($\exists$n on(n), serveAfloor) ; park

Aufzugsbeispiel: Mögliche Aktionen

		•cF	•on	•d_cl	•Poss(a)
•s0		•4	•3, 5	•f	•close
•s1	•do(close, s0)	•4	•3, 5	•t	•open, down(0)…(3), up(5)
•s2	•do(down(3), s1)	•3	•3, 5	•t	•open, down(0)…(2), up(4)…(5)
•s3	•do(open, s2)	•3	•3, 5	•f	•turnoff, close
•s4	•do(turnoff(3), s3)	•3	•5	•f	•close
•s5	•do(close, s4)	•3	•5	•t	•open, down(0)…(2), up(4)…(5)
•s6	•do(up(5), s5)	•5	•5	•t	•open, down(0)…(4)
•s7	•do(open, s6)	•5	•5	•f	•turnoff, close
•s8	•do(turnoff(5), s7)	•5		•f	•close
•s9	•do(close, s8)	•5		•t	•open, down(0)…(4)
•s10	•do(down(0), s9)	•0		•t	•open, up(1)…(5)
•s11	•do(open, s10)	•0		•f	•close

goFloor(n) $=_{def}$?(currentFloor(n)) | ((?(d_closed) | close) ; (up(n) | down(n)))

		•cF	•on	•d_cl	•Do(goFloor(n), s, s')	
•s0		•4	•3, 5	•f		
•s1	•do(close, s0)	•4	•3, 5	•t		•3
•s2	•do(down(3), s1)	•3	•3, 5	•t	•3	
•s3	•do(open, s2)	•3	•3, 5	•f		
•s4	•do(turnoff(3), s3)	•3	•5	•f		
•s5	•do(close, s4)	•3	•5	•t		•5
•s6	•do(up(5), s5)	•5	•5	•t	•5	
•s7	•do(open, s6)	•5	•5	•f		
•s8	•do(turnoff(5), s7)	•5		•f		
•s9	•do(close, s8)	•5		•t		•0
•s10	•do(down(0), s9)	•0		•t	•0	
•s11	•do(open, s10)	•0		•f		

park $=_{def}$ goFloor(0) ; (?(- d_closed) | open)

		•cF	•on	•d_c	•Do(park, s, s')
•s0		•4	•3, 5	•f	
•s1	•do(close, s0)	•4	•3, 5	•t	
•s2	•do(down(3), s1)	•3	•3, 5	•t	
•s3	•do(open, s2)	•3	•3, 5	•f	
•s4	•do(turnoff(3), s3)	•3	•5	•f	
•s5	•do(close, s4)	•3	•5	•t	
•s6	•do(up(5), s5)	•5	•5	•t	
•s7	•do(open, s6)	•5	•5	•f	
•s8	•do(turnoff(5), s7)	•5		•f	
•s9	•do(close, s8)	•5		•t	
•s10	•do(down(0), s9)	•0		•t	•t •t
•s11	•do(open, s10)	•0		•f	•t

serve(n) $=_{def}$ goFloor(n) ; (?(- d_closed) | open) ; turnoff(n)

		•cF	•on	•d_c	•Do(serve(n), s, s')
•s0		•4	•3, 5	•f	
•s1	•do(close, s0)	•4	•3, 5	•t	
•s2	•do(down(3), s1)	•3	•3, 5	•t	•3
•s3	•do(open, s2)	•3	•3, 5	•f	•3
•s4	•do(turnoff(3), s3)	•3	•5	•f	•3 •3
•s5	•do(close, s4)	•3	•5	•t	
•s6	•do(up(5), s5)	•5	•5	•t	•5
•s7	•do(open, s6)	•5	•5	•f	•5
•s8	•do(turnoff(5), s7)	•5		•f	•5 •5
•s9	•do(close, s8)	•5		•t	
•s10	•do(down(0), s9)	•0		•t	
•s11	•do(open, s10)	•0		•f	

while(∃n on(n), serveAfloor)

		•cF	•on	•d_c	•Do(while(∃n on(n), serveAfloor), s, s')
•s0		•4	•3, 5	•f	
•s1	•do(close, s0)	•4	•3, 5	•t	
•s2	•do(down(3), s1)	•3	•3, 5	•t	
•s3	•do(open, s2)	•3	•3, 5	•f	
•s4	•do(turnoff(3), s3)	•3	•5	•f	
•s5	•do(close, s4)	•3	•5	•t	•t •t
•s6	•do(up(5), s5)	•5	•5	•t	•t •t
•s7	•do(open, s6)	•5	•5	•f	•t •t
•s8	•do(turnoff(5), s7)	•5		•f	•t •t
•s9	•do(close, s8)	•5		•t	
•s10	•do(down(0), s9)	•0		•t	
•s11	•do(open, s10)	•0		•f	

control $=_{def}$ while(∃n on(n), serveAfloor) ; park

		•cF	•on	•d_c	•Do(control, s, s')
•s0		•4	•3, 5	•f	
•s1	•do(close, s0)	•4	•3, 5	•t	
•s2	•do(down(3), s1)	•3	•3, 5	•t	
•s3	•do(open, s2)	•3	•3, 5	•f	
•s4	•do(turnoff(3), s3)	•3	•5	•f	
•s5	•do(close, s4)	•3	•5	•t	
•s6	•do(up(5), s5)	•5	•5	•t	•t •t
•s7	•do(open, s6)	•5	•5	•f	•t •t
•s8	•do(turnoff(5), s7)	•5		•f	•t •t
•s9	•do(close, s8)	•5		•t	•t •t
•s10	•do(down(0), s9)	•0		•t	•t •t
•s11	•do(open, s10)	•0		•f	•t

$\mathcal{KB} \models$ **Do(control, s0, s)** ?

Init_{EL}

$\forall$ m [on(m, s0) $\Leftrightarrow$ m = 3 $\vee$ m = 5]

$\forall$ m [currentFloor(m, s0) $\Leftrightarrow$ m = 4]

$\neg$d_closed(s0)

gibt es eine Sequenz $\langle a_1, \ldots, a_n \rangle$, so dass

$\mathcal{KB} \models \text{Do}(\text{control}, s_0, \text{do}(\langle a_1, \ldots, a_n \rangle, s_0))$

Ja: $\langle$close, down(3), open, turnoff(3), close, up(5), open, turnoff(5), close, down(0), open$\rangle$