Lösungshinweise zu Kapitel 11

zu Selbsttestaufgabe 11.1 (Effektaxiome)
Positives Effektaxiom für das Paar (`UNSTACK, CLEAR`):

```
Poss(UNSTACK(x,y),s)  ⇒  CLEAR(y,do(UNSTACK(x,y),s))
```

Negatives Effektaxiom für das Paar (`UNSTACK, ON`):

```
Poss(UNSTACK(x,y),s)  ⇒  ¬ ON(x,y,do(UNSTACK(x,y),s))
```

Positives Effektaxiom für das Paar (`UNSTACK, ONTABLE`):

```
Poss(UNSTACK(x,y),s)  ⇒  ONTABLE(x,do(UNSTACK(x,y),s))
```

zu Selbsttestaufgabe 11.2 (Situationskalkül) Von der Anwendung her ist es intuitiv klar, dass `B` immer noch auf `C` liegt, wenn wir vorher nur `A` bewegt haben. Es gilt zwar `ON(B,C,S0)`, aber mit der bisherigen Formalisierung können wir `ON(B,C,do(STACK(A,B),S0))` nicht ableiten.

zu Selbsttestaufgabe 11.4 (Situationskalkül - Warenkontrolle)

1. Anfangssituation S_0:

$$\{\text{F\_ABGEL}(w_1, S_0), \text{F\_ABGEL}(w_3, S_0), \text{ABGEL}(w_4, S_0), \\ \text{RAB}(w_1, S_0), \text{RAB}(w_4, S_0)\}$$

2. (a) Ausführungsbedingungen für die Aktion S_AUS:

$$Poss(\text{S\_AUS}(x), s) \equiv \text{ABGEL}(x, s) \wedge \neg\text{AUSS}(x, s)$$

   (b) Effektaxiome für die Aktion RAB_IERE:

$$Poss(\text{RAB\_IERE}(x), s) \Rightarrow \text{RAB}(x, do(\text{RAB\_IERE}(x), s))$$

   (c) Rahmenaxiom für die Aktion S_AUS für das Fluent AUSS:

$$\neg\text{AUSS}(x, s) \wedge x \neq y \Rightarrow \neg\text{AUSS}(x, do(\text{S\_AUS}(y), s))$$

3. Zielbeschreibung:

$$\exists t\ \text{RAB}(w_3, t) \wedge \text{AUSS}(w_4, t)$$

4. Endsituation S_2, in der das Ziel ausgehend von der Anfangssituation S_0 erreicht wird:

$$\begin{aligned} S_1 &= do(\text{RAB\_IERE}(w_3), S_0) \\ S_2 &= do(\text{S\_AUS}(w_4), S_1) \end{aligned}$$

zu Selbsttestaufgabe 11.3 (Rahmenaxiome) Ein Block ist auch noch nach einer `STACK`-Operation frei, wenn kein anderer Block auf ihn gestapelt wurde:

$$\texttt{CLEAR(x,s)} \wedge \texttt{x} \neq \texttt{v} \quad \Rightarrow \quad \texttt{CLEAR(x,do(STACK(u,v),s))}$$

Ein Block ist auch noch nach einer `STACK`-Operation nicht frei, wenn er dies vorher nicht war, da diese Operation nur Blöcke, die auf dem Tisch stehen, stapeln kann:

$$\neg\, \texttt{CLEAR(x,s)} \quad \Rightarrow \quad \neg\, \texttt{CLEAR(x,do(STACK(u,v),s))}$$

zu Selbsttestaufgabe 11.5 (`R-STRIPS`) Im Folgenden ist ein möglicher Programmablauf von `R-STRIPS` skizziert. Dabei kennzeichnen die hochgestellten Zahlen hinter einer Programmvariablen den aktuellen Schleifendurchlauf und die Rekursionstiefe. Beispielsweise bezieht sich $\texttt{g}^{2.1}$ auf den Wert von `g` im 1. Schleifendurchlauf des rekursiven `R-STRIPS`-Aufrufs, der im 2. Schleifendurchlauf des ursprünglichen `R-STRIPS`-Aufrufs erzeugt wurde.

R-STRIPS(G, Start)
1. P^1 := [ ]
2. S^1 := {CLEAR(B), ON(B,A), ON(A,C), ONTABLE(C)}
3. Es gilt $G \not\subseteq S^1$
4. g^1 := ON(A,B) % Es könnte auch ON(B,C) gewählt werden
5. Op^1 := MOVE(A,C,B) % Es könnte auch STACK(A,B) gewählt werden
6. R-STRIPS({ON(A,C), CLEAR(A), CLEAR(B)}, S^1)
   % Vorbedingung von MOVE(A,C,B) ist
   % {ON(A,C), CLEAR(A), CLEAR(B)}
   1. $P^{1.1}$:= [ ]
   2. $S^{1.1}$:= S^1
   3. Es gilt {ON(A,C), CLEAR(A), CLEAR(B)} $\not\subseteq S^{1.1}$
   4. $g^{1.1}$:= CLEAR(A) % hier einzige Wahlmöglichkeit
   5. $Op^{1.1}$:= UNSTACK(B,A)
   6. R-STRIPS({ON(B,A), CLEAR(B)}, $S^{1.1}$)
      % Vorbedingung von UNSTACK(B,A) ist
      % {ON(B,A), CLEAR(B)}
      1. $P^{1.1.1}$:= [ ]
      2. $S^{1.1.1}$:= $S^{1.1}$
      3. Es gilt {ON(B,A), CLEAR(B)} $\subseteq S^{1.1.1}$ % ($S^{1.1.1}$ = Start)
      9. return([ ])
   6. $P_C^{1.1}$:= [ ]
   7. $S^{1.1}$:= UNSTACK(B,A) ($S^{1.1}$)
      = {ON(A,C), ONTABLE(B), ONTABLE(C), CLEAR(A), CLEAR(B)}
   8. $P^{1.1}$ = $P + P_C^{1.1} + [Op^{1.1}]$
      = [ ] + [ ] + [UNSTACK(B,A)]
      = [UNSTACK(B,A)]
   3. Es gilt {ON(A,C), CLEAR(A), CLEAR(B)} $\subseteq S^{1.1}$
   9. return(UNSTACK(B,A))
6. P_C^1 := [UNSTACK(B,A)]
7. S^1 := Op^1 ($P_C^1(S^1)$)
   = MOVE(A,C,B)([UNSTACK(B,A)](S^1))
   = {ON(A,B), ONTABLE(B), ONTABLE(C), CLEAR(A), CLEAR(C)}
8. P^1 := $P^1 + P_C^1 + [Op^1]$
   = [UNSTACK(B,A), MOVE(A,C,B)]
3. In S^1 ist Ziel ON(B,C) noch nicht erfüllt
4. g^2 := ON(B,C)
5. Op^2 := STACK(B,C)
6. R-STRIPS ({ONTABLE(B), CLEAR(B), CLEAR(C)}, S^1)
   1. $P^{2.1}$:= [ ]
   2. $S^{2.1}$:= S^1
   3. CLEAR(B) $\notin S^{2.1}$
   4. $g^{2.1}$:= CLEAR(B)
   5. $Op^{2.1}$:= UNSTACK(A,B)

6. R-STRIPS ({ON(A,B), CLEAR(A)}, $S^{2.1}$)
   % Vorbedingung für $Op^{2.1}$ erzeugen
   1. $P^{2.1.1}$:= [ ]
   2. $S^{2.1.1}$:= $S^{2.1}$
   3. Es gilt {ON(A,B), CLEAR(A)} $\subseteq S^{2.1.1}$
   9. return ([ ])
6. $P_C^{2.1}$:= [ ]
7. $S^{2.1}$:= UNSTACK(A,B)($S^{2.1}$)
   = {ONTABLE(A), ONTABLE(B), ONTABLE(C), CLEAR(A), CLEAR(B), CLEAR(C)}
8. $P^{2.1}$:= [UNSTACK(A,B)]
3. Es gilt {ONTABLE(B), CLEAR(B), CLEAR(C)} $\subseteq S^{2.1}$
9. return([UNSTACK(A,B)])

6. P_C^2 := [UNSTACK(A,B)]
7. S^2 := STACK(B,C) (UNSTACK(A,B)(S^1))
   = {ON(B,C), ONTABLE(A), ONTABLE(C), CLEAR(A), CLEAR(B)}
8. P^2 := $P^1 + P_C^2 + [Op^2]$
   = [UNSTACK(B,A), MOVE(A,C,B), UNSTACK(A,B), STACK(B,C)]
3. In S^2 ist das Ziel ON(A,B) nicht erfüllt
   % obwohl ON(A,B) vorher schon einmal erzielt wurde!
4. g^3 := ON(A,B)
5. Op^3 := STACK(A,B)
6. R-STRIPS ({ONTABLE(A), CLEAR(A), CLEAR(B)}, S^2)
   ... % alle Bedingungen sind in S^2 erfüllt
   9. return ([ ])
6. P_C^3 := [ ]
7. S^3 := STACK(A,B)(S^2)
   = {ON(A,B), ON(B,C), ONTABLE(C), CLEAR(A)}
8. P^3 := [UNSTACK(B,A), MOVE(A,C,B), UNSTACK(A,B), STACK(B,C), STACK(A,B)]
3. Es gilt {ON(A,B), ON(B,C)} $\subseteq S^3$
9. return (P^3)

zu Selbsttestaufgabe 11.6 (Türme von Hanoi)

1. Wir verwenden die folgenden Prädikate:

TOP(x,t)	:	x ist die oberste Scheibe auf Stab t
BOTTOM(x,t)	:	x ist die unterste Scheibe auf Stab t
ON(x,y)	:	Die Scheibe x liegt unmittelbar auf Scheibe y
LT(x,y)	:	Die Scheibe x ist kleiner als die Scheibe y
EMPTY(t)	:	Auf dem Stab t liegt keine Scheibe

2. Die initiale STRIPS-Datenbasis enthält folgende Elemente:

$$\begin{aligned} & \{\texttt{TOP}(D_1,\texttt{A}), \texttt{ON}(D_1,D_2), \texttt{ON}(D_2,D_3),\ldots,\texttt{ON}(D_{n-1},D_n)\} \\ \cup\ & \{\texttt{BOTTOM}(D_n,\texttt{A}),\texttt{EMPTY(B)},\texttt{EMPTY(C)}\} \\ \cup\ & \{\texttt{LT}(D_i,D_j) \mid i<j\} \end{aligned}$$

Als Zielbeschreibung erhalten wir:

$\{$TOP(D_1,C),
ON(D_1,D_2), ON(D_2,D_3),...,ON(D_{n-1},D_n),
BOTTOM(D_n,C)$\}$

3. Wir definieren vier verschiedene STRIPS-Operatoren, die Spielzüge in jeweils unterschiedlichen Spielsituationen realisieren:

MOVE_LAST_TO_EMPTY(t,t')	:	Lege die einzige Scheibe, die sich auf t befindet, auf den (leeren) Stab t'.
MOVE_TO_EMPTY(t,t')	:	Lege die oberste Scheibe von Stab t (auf dem sich noch mindestens eine weitere Scheibe befindet) auf den (leeren) Stab t'.
MOVE_LAST(t,t')	:	Lege die einzige Scheibe, die sich auf t befindet, auf den (nicht leeren) Stab t'.
MOVE(t,t')	:	Lege die oberste Scheibe von Stab t (auf dem sich noch mindestens eine weitere Scheibe befindet) auf den (nicht leeren) Stab t'.

MOVE_LAST_TO_EMPTY(t,t'):	C:	TOP(x,t), BOTTOM(x,t), EMPTY(t')
	D:	TOP(x,t), BOTTOM(x,t), EMPTY(t')
	A:	TOP(x,t'), BOTTOM(x,t'), EMPTY(t)
MOVE_TO_EMPTY(t,t'):	C:	TOP(x,t), ON(x,y), EMPTY(t')
	D:	TOP(x,t), ON(x,y), EMPTY(t')
	A:	TOP(x,t'), BOTTOM(x,t'), TOP(y,t)
MOVE_LAST(t,t'):	C:	TOP(x,t), BOTTOM(x,t), TOP(y,t'), LT(x,y)
	D:	TOP(x,t), BOTTOM(x,t), TOP(y,t')
	A:	TOP(x,t'), ON(x,y), EMPTY(t)
MOVE(t,t'):	C:	TOP(x,t), ON(x,y), TOP(z,t'), LT(x,z)
	D:	TOP(x,t), ON(x,y), TOP(z,t')
	A:	TOP(x,t'), ON(x,z), TOP(y,t)

4. Bei $n = 3$ erhalten wir folgende STRIPS-Startdatenbasis:

$\{$ TOP(D_1,A),
ON(D_1,D_2), ON(D_2,D_3),
BOTTOM(D_3,A),
EMPTY(B),
EMPTY(C),
LT(D_1,D_2), LT(D_2,D_3), LT(D_1,D_3)$\}$

Die Zielbeschreibung enthält:

{ TOP(D_1,C),
ON(D_1,D_2), ON(D_2,D_3).
BOTTOM(D_3,C)}

Bei günstiger Auswahl des als nächstes zu bearbeitenden Teilziels und günstiger Auswahl einer Operatorinstanz zur Erreichung dieses Teilziels erhalten wir folgenden (optimalen) Plan:

MOVE_TO_EMPTY(A,C),
MOVE_TO_EMPTY(A,B),
MOVE_LAST(C,B),
MOVE_LAST_TO_EMPTY(A,C),
MOVE_TO_EMPTY(B,A),
MOVE_LAST(B,C),
MOVE_LAST(A,C)

Alternative Lösung:
Alternativ kann man die Stäbe auch als Scheiben A, B und C modellieren, die größer als alle anderen, untereinander gleichgroß und daher unbeweglich sind. Das vereinfacht die Lösung stark, denn man kommt nun mit drei Prädikaten aus:

LT(x,y)	:	Die Scheibe x ist kleiner als die Scheibe y
ON(x,y)	:	Scheibe x liegt auch Scheibe y
EMPTY(x)	:	Auf der Scheibe x liegt keine weitere Scheibe.

Als Startzustand ergibt sich dann:

$\{$EMPTY(D_1), ON(D_1,D_2), ON(D_2,D_3),$\ldots$,ON(D_{n-1},D_n), ON(D_n,A),EMPTY(B),EMPTY(C)$\}$
$\cup$ $\{$LT(D_i,D_j) $\mid i < j\}$
$\cup$ $\{$LT(D_i,X) $\mid i = 1,\ldots,n, X \in \{A,B,C\}\}$

Der Zielzustand wird beschrieben durch

$\{$EMPTY(D_1),ON(D_1,D_2),ON(D_2,D_3),$\ldots$,ON(D_{n-1},D_n),ON(D_n,C),EMPTY(A),EMPTY(B)$\}$

Leere Stäbe und unterste Scheiben müssen jetzt nicht extra berücksichtigt werden und man kommt mit einer einzigen Verschiebeoperation aus, bei der die Scheibe x von der Scheibe y auf die Scheibe z gelegt wird:

MOVE(x,y,z): C: ON(x,y),EMPTY(x),EMPTY(z),LT(x,z)
D: ON(x,y),EMPTY(z)
A: ON(x,z),EMPTY(y)

Als Plan für den Fall $n = 3$ ergibt sich entsprechend:

MOVE(D_1,D_2,C),
MOVE(D_2,D_3,B),
MOVE(D_1,C,D_2),
MOVE(D_3,A,C),
MOVE(D_1,D_2,A),
MOVE(D_2,B,D_3),
MOVE(D_1,A,D_2)

zu Selbsttestaufgabe 11.7 (Sokoban)

1. Zunächst nummerieren wir das Spielfeld wie in Abbildung L.17 angegeben.

   Um die möglichen Bewegungen des Lagerarbeiters zu beschreiben, geben wir die Nachbarschaftsbeziehungen zwischen den Spielfeldern an:

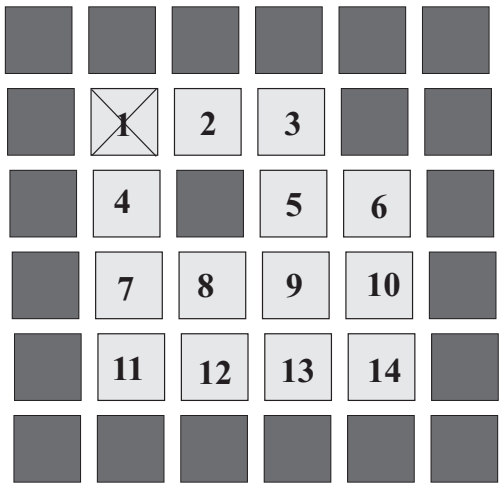
$$\begin{aligned} NB_1 := \{&n(1,2), n(1,4), n(2,3), n(3,5), n(4,7), n(5,6), n(5,9), n(6,10),\\ &n(7,8), n(7,11), n(8,9), n(8,12), n(9,10), n(9,13), n(10,14),\\ &n(11,12), n(12,13), n(13,14)\} \end{aligned}$$

$$NB_2 := \{(x,y) |\ (y,x) \in NB_1\}$$

$$NB \ := NB_1 \cup NB_2$$

   Nun können wir „Schiebelinien“ bestimmen: Dies sind drei Felder, die senkrecht oder waagerecht aneinandergrenzen, beispielsweise die Felder 7, 8 und 9. Die Kiste kann nur dann geschoben werden, wenn sie in der Mitte einer solchen Schiebelinie steht. In unsere Datenbasis nehmen wir also alle Schiebelinien (sl) auf:

$$\begin{aligned} SL := \{&sl(1,2,3), sl(3,2,1), sl(1,4,7), sl(7,4,1), sl(4,7,11), sl(11,7,4),\\ &sl(7,8,9), sl(9,8,7) sl(8,9,10), sl(10,9,8), (11,12,13), sl(13,12,11),\\ &sl(12,13,14), sl(14,13,12), sl(3,5,9), sl(9,5,3), sl(5,9,13),\\ &sl(13,9,5), sl(6,10,14), sl(14,10,6)\} \end{aligned}$$



Beschriftung der Felder der dargestellten Sokoban-Situation.

Abbildung L.17 Nummerierung des Felder (Selbsttestaufgabe 11.7)

Die Nachbarschaftbeziehungen und die Schiebelinien ändern sich im Laufe des Spiels nicht. Dass die Kiste oder der Lagerarbeiter auf einem Feld stehen, beschreiben wir mit den einstelligen Prädikaten *kiste* und *arbeiter*, dass ein Feld unbesetzt ist, mit dem einstelligen Prädikat *empty*.

Damit können wir nun den Start- und den Zielzustand beschreiben:

Der Startzustand ist die Datenbasis

$$NB \cup SL \cup \{kiste(5), arbeiter(6)\}$$

und die Menge der Zielzustände besteht aus allen Datenbasen, die das Literal $kiste(1)$ enthalten.

Wir definieren nun zwei `STRIPS`-Operatoren *go* und *push*.

$go(x, y)$ ist erklärt durch

$$\begin{array}{ll} C: & nb(x,y), arbeiter(x), empty(y) \\ D: & arbeiter(x), empty(y) \\ A: & arbeiter(y), empty(x) \end{array}$$

$push(x, y, z)$ ist erklärt durch

$$\begin{array}{ll} C: & sl(x,y,z), arbeiter(x), kiste(y), empty(z) \\ D: & arbeiter(x), kiste(y), empty(z) \\ A: & arbeiter(y), empty(x), kiste(z) \end{array}$$

(Die Bedingung $empty(z)$ wird für ein Spiel mit nur einer Kiste nicht benötigt.)

2. Beim Top-down-Ansatz starten wir mit dem Zustand

$$arbeiter(6),\ kiste(5)$$

Der sich damit ergebende Suchraum ist in Abbildung L.18 skizziert. Die Kanten des Baumes in Abbildung L.18 sind mit den in den jeweiligen Zuständen möglichen Operationen beschriftet; die Knoten sind die jeweiligen Folgezustände.

Eine Bottom-up-Suche würde von den Zielzuständen $arbeiter(n), kiste(1)$ erfolgen, wobei n zunächst beliebig ist, jedoch die Betrachtung der Werte 2 und 4 genügt (wurde die Kiste auf das Feld 1 geschoben, steht der Arbeiter auf einem der Felder 2 oder 4). Von dort aus würden die Züge betrachtet, die zu diesen Zuständen führen, und auf diese Weise würde versucht, zum Startzustand zu gelangen.

3. In Abbildung L.19 ist eine Startsituation mit zwei Kisten dargestellt, bei der es nicht möglich ist, zum Ziel zu kommen, in dem man erst die eine und dann die andere Kiste auf ein Zielfeld schiebt. Tatsächlich muss man bei der Zielsuche stets beide Kisten im Auge haben (was den eigentlichen Reiz des Spiels ausmacht).

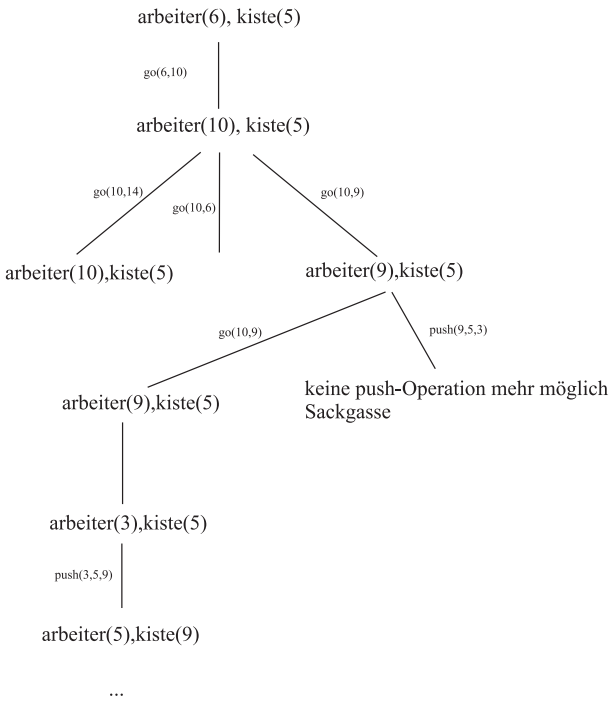
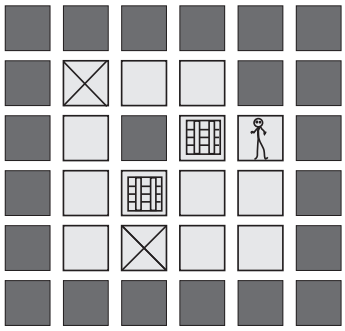
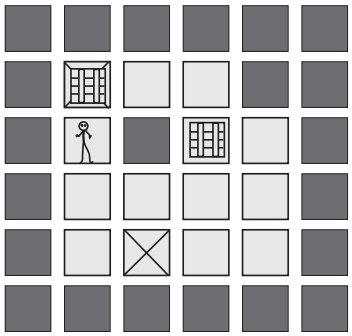


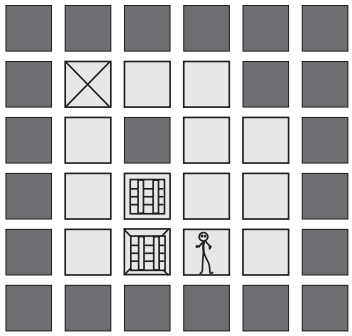
Abbildung L.18 Darstellung des Suchraums (Selbsttestaufgabe 11.7)



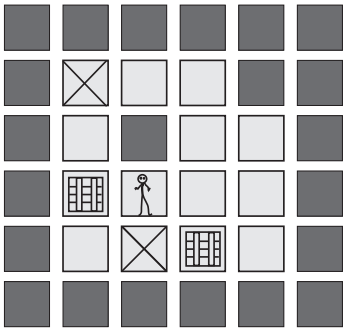
Startzustand eines Sokoban-Spiels mit zwei Kisten.



Ein Teilziel ist erreicht: Eine Kiste ist auf einem der Zielfelder. Nun ist es jedoch nicht möglich, die zweite Kiste zu schieben.



Auch hier ist ein Teilziel erreicht, jedoch kann das Gesamtziel nicht erreicht werden.



Zielführend ist, die beiden Kisten abwechselnd zu bewegen, so dass sie in diese Position kommen.

Abbildung L.19 Sussman-Anomalie (Selbsttestaufgabe 11.7)

zu Selbsttestaufgabe 11.8 (STRIPS - Gebäudereinigung)

1. TRANSPORT(b_1, b_2):
   C: ROB_BER(b_1), EIMER_BER(b_1)
   D: ROB_BER(b_1), EIMER_BER(b_1)
   A: ROB_BER(b_2), EIMER_BER(b_2)
2. FAHREN($EG.2, EG.1$), TRANSPORT($EG.1, UG.3$), PUTZEN($EG.3$)
3. PUTZE_BER(b_1, b_2, b_3):
   C: ROB_BER(b_2), EIMER_BER(b_3), ZIEL_BER(b_1)
   D: ROB_BER(b_2), EIMER_BER(b_3) (, ZIEL_BER(b_1))
   A: ROB_BER(b_1), EIMER_BER(b_1), FERTIG()

zu Selbsttestaufgabe 11.9 (SMODELS**)** Im Folgenden nehmen wir an, dass S eine Antwortmenge zu dem in den Abbildungen 11.17 - 11.19 gegebenen Programm sei, die die jeweils angegebenen Literale enthalte, und führen dies zum Widerspruch.

1. Da im Initialzustand

   $$\texttt{on}(5,6,0),\ \texttt{block}(6),\ \texttt{block}(5),\ \texttt{location}(3),\ \texttt{time}(0),\ 0 < \texttt{lasttime}$$

   vorhanden ist, ist mit $\texttt{move}(6,3,0)$ das zweite Constraint aus Abbildung 11.18 verletzt (ein Block, der nicht frei ist – in diesem Fall Block 6 –, kann nicht bewegt werden).

2. Da kein Regelkopf des Programms negierte move-Atome enthält, kann eine Antwortmenge nicht -move(6,3,0) enthalten (vgl. Proposition 9.37).

3. Da wegen des ersten Literals move(5,3,0) auch $\texttt{on}(5,3,1) \in S$ gelten muss, würde dies gemeinsam mit dem zweiten Literal move(3,table,1) im Konflikt mit dem zweiten Constraint aus Abbildung 11.18 stehen.

4. Eine Antwortmenge kann definitionsgemäß nur Literale, nicht jedoch eine Default-Negation wie not on(2,1,0) enthalten.

5. Da $\texttt{move}(5,6,2) \in S$ gilt, folgt mit der ersten DEFINE-Regel aus Abbildung 11.17, dass auch $\texttt{on}(5,6,3) \in S$ gelten muss. Da S damit die Literale

   $$\texttt{on}(5,6,3), \texttt{block}(5), \texttt{location}(6), \texttt{time}(3)$$

   enthält und weiterhin $6 \neq 4$ gilt, muss mit der dritten DEFINE-Regel aus Abbildung 11.17 ("Eindeutigkeit des Ortes") auch $-\texttt{on}(5,4,3) \in S$ gelten. Wegen $\texttt{lasttime} = 3$ wäre damit aber das fünfte TEST-Constraint

   $$\texttt{:- not on}(5,4,\texttt{lasttime}).$$

   aus Abbildung 11.19 verletzt.

6. S muss wegen der sechsten DEFINE-Regel aus Abbildung 11.19 on(6,table,0) enthalten. Würde es zusätzlich auch das Literal -on(6,table,0) enthalten, wäre S nicht konsistent, was eine Antwortmenge aber sein muss.