Aufgabenblatt 4 – Musterlösung

Aufgabe 1 – Planung in der erweiterten Blockwelt (30 %)

a) Wir nummerieren die Bausteine folgendermaßen:

Prädikate (atemporal):

block/1	: Typprädikat
dach/1	: Typprädikat
größe/1	: Typprädikat, definiert gültige Größen
größe/2	: Größe eines Objektes
farbe/2	: Farbe eines Objektes
n/2	: Nachfolgerprädikat auf natürlichen Zahlen, wird für Höhe verwendet
ge/2	: größer-gleich-Relation (falls mehr als nur 2 Größen vorhanden sind)

Prädikete (temporal):

v/1	: bezeichnet ein verfügbares Bauteil (eines, das noch nicht verbaut ist)
höhe/1	: aktuelle Höhe des zu bauenden Hauses
etage/2	: Baustein im Haus mit Etage und Bausteinnummer

Bemerkungen:

- Die Verwendung von größe/1 und größe/2 ist erlaubt. Durch die unterschiedliche Arität handelt es sich um verschiedene Prädikate.
- Statt v/1 man könnte alternativ auch einen Ort "Kiste" als Konstante definieren und beispielsweise in/2 verwenden. Die Blocknummer ist wichtig, weil mehrere gleiche Blöcke zu unterscheiden sind – dasselbe gilt für Dächer. Dächer haben keine Farbe, weil diese beim Häuserbauen keine Rolle spielt.
- Entweder das Zählen von Etagen oder aber ein Prädikat bottom/1 ist notwendig, denn ansonsten würden auch höhere Häuser eine (wie auch immer korrekt beschriebene) Zielbeschreibung erfüllen.
- Man könnte auch mehrere Informationen in *ein* block/3-Prädikat einkodieren (ID, Farbe, Größe), anstatt separate Prädikate größe/2 und farbe/2 für jeden Block zu nutzen.

Konstanten:

S,G,W	: Farben
1,2	: Größen (klein, groß)
1,...9	: IDs für Bausteine

Bemerkung:

- 1 und 2 sind sowohl Größen, als auch IDs. Diese „Überladung“ von Konstanten bringt hier keine Probleme. Bei dieser Modellierung von Größen mittels Zahlen in Verbindung mit der größer-gleich-Relation kann man auch weitere Blockgrößen einführen.

Typische Fehler: nicht offen modelliert, free oder available fehlt.

b)
S_0: { block(1), block(2), ..., block(6), dach(7), dach(8), dach(9),

größe(1,1), größe(2,1), ..., größe(6,2), größe(7,1), größe(8,1), größe(9,2),
farbe(1,S), farbe(2,S), ..., farbe(6,W),
farbe(S), farbe(G), farbe(W), größe(1), größe(2), n(0,1), n(1,2), n(2,3),
höhe(0),
ge(1,1), ge(2,1), ge(2,2)
v(1),v(2),...,v(9) }

Typischer Fehler: deutlich unvollständiger Startzustand.

c)
S_Z:{etage(1,x),etage(2,y),etage(3,z),dach(z)}
alternativ: {höhe(3),etage(3,x),dach(x)}

Typische Fehler: unnötige Details im Zielzustand verhindern (potentielle) Lösungen

d)

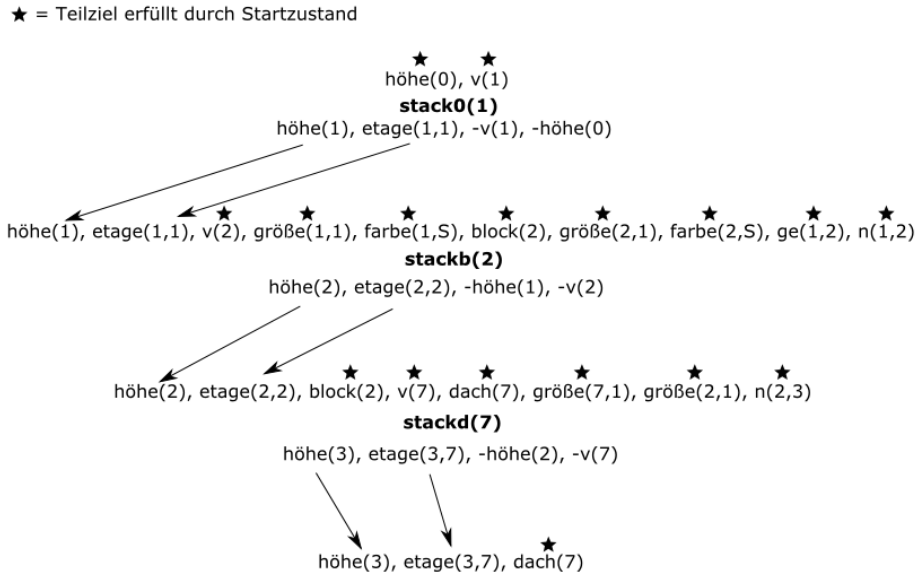
ACT: stack0(x) // ein beliebiges Bauteil als 1. Etage verbauen
PRE: höhe(0), v(x)
EFF: höhe(1), etage(1,x), ¬v(x), ¬höhe(0)
Bemerkung: höhe/1 und etage/2 werden aktualisiert, die Verfügbarkeit des Bauteils entfernt.

ACT: stackb(x) // einen verfügbaren Block auf einen anderen Block legen
PRE: höhe(h), etage(h,b), v(x), größe(b,g), farbe(b,f), block(x), größe(x,gx), farbe(x,f), ge(g,gx), n(h,h1)
EFF: höhe(h1), etage(h1,x), ¬höhe(h), ¬v(x)
Bemerkung: die Überprüfung der Farbe stellt sicher, dass b kein Dach sein kann.

ACT: stackd(x) // ein verfügbares, passendes Dach auf die aktuelle Etage legen
PRE: höhe(h), etage(h,b), block(b), v(x), dach(x), größe(x,g), größe(b,g), n(h,h1)
EFF: höhe(h1), etage(h1,x), ¬höhe(h), ¬v(x)

Typische Fehler: Prädikat im Argument, Rechnen in PRE, fehlende Typisierung in PRE, ODER oder NOT verwendet.

e)
Lösungsplan: stack0(1),stackb(2),stackd(7). POP:



1. Beginne mit dem leeren Plan, in dem zwei virtuelle Aktionen A0 den Startzustand und AZ das Ziel repräsentieren. A0 mit leerer PRE und DEL generiert die Literale von S0, AZ mit leerer ADD und DEL generiert mittels seiner Vorbedingungen das Ziel SZ. O={A0<AZ}
2. Belegung von z = 7 in AZ (dach(7) erfüllt in ADD von A0)
3. Erfüllen des Teilziels etage(3,7) durch eine stackd-Aktion (A1), dabei findet Substitution {x/7} statt. Die PRE der Aktion A1 definiert die neuen Teilziele (höhe(h), etage(h,b), block(b), v(7), dach(7), größe(7,g), größe(b,g), n(h,3)). (v(7) und dach(7) erfüllt in ADD von A0)
4. Belegung von g = 1 in A1 (größe(7,1) erfüllt in ADD von A0)
5. Belegung von h = 2 in A1 (n(2,3) und höhe(2) erfüllt in ADD von A0)
6. Belegung von b = 2 in A1 (block(2) und größe(2,1) erfüllt in ADD von A0)
7. Erfüllen des Teilziels etage(2,2) durch eine stackb-Aktion (A2), dabei findet Substitution {x/2} statt. Die PRE der Aktion A2 definiert die neuen Teilziele (höhe(h), etage(h,b), v(2), größe(b,g), farbe(b,f), block(2), größe(2,gx), farbe(x,f), ge(g,gx), n(h,2)). (v(2) und block(2) erfüllt in ADD von A0)
8. Belegung von gx = 1 in A2 (größe(2,1) erfüllt in ADD von A0)
9. Belegung von h = 1 in A2 (n(1,2) und höhe(1) erfüllt in ADD von A0)
10. Belegung von g = 1 in A2 (ge(1,1) erfüllt in ADD von A0)
11. Belegung von f = S in A2 (farbe(2,S) erfüllt in ADD von A0)
12. Belegung von b = 1 in A2 (größe(1,1) und farbe(1,S) erfüllt in ADD von A0)
13. Erfüllen des Teilziels etage(1,1) durch eine stack0-Aktion (A3), dabei findet Substitution {x/1} statt. Die PRE der Aktion A3 definiert die neuen Teilziele (höhe(0), v(1)}. (höhe(0) und v(1) erfüllt in ADD von A0).

Aufgabe 2 – Rückwärtsplanung (15 %)

0. [Z(1)]
1. a(1,0); [A(1,0), B(1)]
1. keine relevante&konsistente Aktion existiert, denn B(1) wird nicht generiert. Backtracking zu 0. auslösen.
2. a(1,1); [A(1,1), B(1)]
2.1 keine relevante&konsistente Aktion existiert, denn B(1) wird nicht generiert. Backtracking zu 0. auslösen.
3. b1(1); [K(1)]
3.1 b2(1); [L(1)]
3.2 b3(1,0); [A(1,0),K(1),Z(1)] Zyklus, denn {K(1)} in 3. ist Teilmenge der aktuellen Zielmenge (gilt übrigens auch für {Z(1)} in 0.). Backtracking zu 0. auslösen.
4. c(x); {A(1,0),B(0),C(0)} Ziel erfüllt

Aufgabe 3 – Vorwärtsplanung mit A* **(30 %)**

A*-Suchbaum. Unter/an den Knoten stehen die f-Werte (g+h). Der erste Pfad (links) führt in einen Zyklus. Danach stehen die Operatoren mv3(C,B) und mv1(C) zur Wahl, wobei der Tiebreak durch mv3 < mv1 gemäß Aufgabenstellung aufgelöst wird. Der Nachfolger von mv3 hat einen f-Wert von 4. Also wird mv1(C) als nächstes expandiert. Dieser Knoten hat 4 Nachfolger, 2 davon mit f-Wert von jeweils 3. Beides sind mv2-Aktionen, davon ist mv2(A,B) die lexikographisch kleinste. Mit dieser Aktion landen wir in einem mit 4 bewerteten Knoten. Der nächste Expandierungsschritt führt dann zur Lösung. Nicht dargestellt sind Knoten, die Zyklen darstellen würden oder aufgrund von dynamischer Programmierung entfallen.