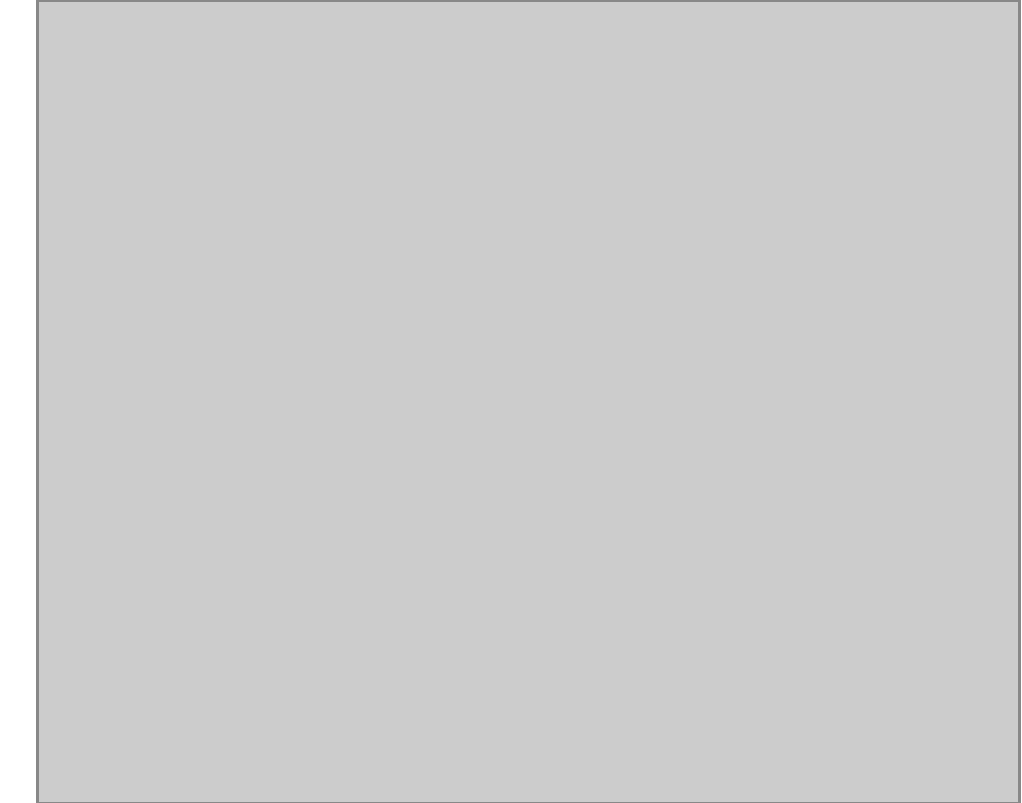


```

1 % Die Schnittstelle umfasst
2 %   start_description   ;Beschreibung des Startzustands
3 %   start_node         ;Test, ob es sich um einen Startknoten
   handelt
4 %   goal_node          ;Test, ob es sich um einen Zielknoten
   handelt
5 %   state_member       ;Test, ob eine Zustandsbeschreibung in
   einer Liste
6 %                       von Zustandsbeschreibungen enthalten ist
7 %   expand              ;Berechnung der Kind-Zustandsbeschreibungen
8 %   eval-path          ;Bewertung eines Pfades

```



```

9
10
11 start_description([
12   block(block1),
13   block(block2),
14   block(block3),
15   block(block4), %mit Block4
16   on(table,block2),
17   on(table,block3),
18   on(block2,block1),
19   on(table,block4), %mit Block4
20   clear(block1),

```

```

1 %Die Schnittstelle umfasst
2 %   start_description   ;Beschreibung des Startzustands
3 %   start_node         ;Test, ob es sich um einen Startknoten handelt
4 %   goal_node          ;Test, ob es sich um einen Zielknoten handelt
5 %   state_member       ;Test, ob eine Zustandsbeschreibung in einer Liste
6 %                       von Zustandsbeschreibungen enthalten ist
7 %   expand              ;Berechnung der Kind-Zustandsbeschreibungen
8 %   eval-path          ;Bewertung eines Pfades

```

```

9 /*
10 %<3 Blöcke>
11 start_description([
12   block(block1),
13   block(block2),
14   block(block3),
15   on(table,block2),
16   on(table,block3),
17   on(block2,block1),
18   clear(block1),
19   clear(block3),
20   handempty
21 ]).
22 goal_description([
23   block(block1),
24   block(block2),
25   block(block3),
26   on(table,block3),
27   on(table,block1),
28   on(block1,block2),
29   clear(block3),
30   clear(block2),
31   handempty
32 ]).
33 %</3 Blöcke>
34 */

```

```

35
36 %<4 Blöcke>
37 start_description([
38   block(block1),
39   block(block2),
40   block(block3),
41   block(block4),
42   on(table,block2),
43   on(table,block3),
44   on(block2,block1),
45   on(table,block4),
46   clear(block1),

```

```

21 clear(block3),
22 clear(block4), %mit Block4
23 handempty
24 ]).
25
26 goal_description([
27     block(block1),
28     block(block2),
29     block(block3),
30     block(block4), %mit Block4
31     on(block4,block2), %mit Block4
32     on(table,block3),
33     on(table,block1),
34     on(block1,block4), %mit Block4
35 % on(block1,block2), %ohne Block4
36     clear(block3),
37     clear(block2),
38     handempty
39 ]).
40
41
42
43 start_node((start,_,_)).
44
45 goal_node( (_,State,_)):-
46     "Zielbedingungen einlesen"
47     "Zustand gegen Zielbedingungen testen".
48
49
50
51 % Aufgrund der Komplexität der Zustandsbeschreibungen kann
52 state_member nicht auf
53 %
54 state_member(_,[]):- !,fail.
55
56 state_member(State,[FirstState|_]):-
57     "Test, ob State bereits durch FirstState beschrieben war. Tipp:
58     Eine
59     Lösungsmöglichkeit besteht in der Verwendung einer
60     Mengenoperation, z.B. subtract" ,!.
61
62 %Es ist sichergestellt, dass die beiden ersten Klauseln nicht
63 zutreffen.
64
65 state_member(State,[_|RestStates]):-

```

```

47 clear(block3),
48 clear(block4),
49 handempty
50 ]).
51
52 goal_description([
53     block(block1),
54     block(block2),
55     block(block3),
56     block(block4),
57     on(block4,block2),
58     on(table,block3),
59     on(table,block1),
60     on(block1,block4),
61     clear(block3),
62     clear(block2),
63     handempty
64 ]).
65 %</4 Blöcke>
66
67
68
69
70 start_node((start,_,_)).
71
72 goal_node( (_,State,_)):-
73     %used TODO:"Zielbedingungen einlesen"
74     goal_description(Goal),
75     %used TODO:"Zustand gegen Zielbedingungen testen"
76     state_member(State,[Goal]).
77
78
79 % Aufgrund der Komplexität der Zustandsbeschreibungen kann state_member
80 nicht auf
81 %
82 state_member(_,[]):- !,fail.
83
84 state_member(State,[FirstState|_]):-
85     %used TODO:"Test, ob State bereits durch FirstState beschrieben war. Tipp:
86     Eine Lösungsmöglichkeit besteht in der Verwendung einer Mengenoperation,
87     z.B. subtract" ,!.
88     mysubset(State,FirstState).
89
90 %Es ist sichergestellt, dass die beiden ersten Klauseln nicht zutreffen.
91
92 state_member(State,[_|RestStates]):-

```

```

62 "rekursiver Aufruf".
63
64
65 eval_path([(_,State,Value)|RestPath]):-
66     eval_state(State,"Rest des Literals bzw. der Klausel"
67     "Value berechnen".
68
69

```

```

90 %used TODO:"rekursiver Aufruf".
91 state_member(State,RestStates).
92
93 %<TEIL 2>
94 %eval_path([(X,State,Value)|RestPath]):-
95     eval_path(Path):-
96         %used TODO:eval_state(State,"Rest des Literals bzw. der Klausel"
97         length(Path,G),
98         eval_state(Path,G).
99
100 %<Suchverfahren> //Ein Algorithmus auswählen
101
102 %used TODO: A-Algorithmus
103 /* eval_state([(X,State,Value)|_],G) :-
104     heuristik(State,H),
105     F is G + H, %A-Algorithmus: f(n) = g(n) + h(n)
106     %nl,write(['Moeglichkeit: ', X, ' Kosten: ', F]), %ToDebug
107     Value = F.
108 */
109
110 %used TODO: gierige Bestensuche
111 eval_state([(X,State,Value)|_],_) :-
112     heuristik(State,H),
113     %nl,write(['Moeglichkeit: ', X, ' Kosten: ', H]), %ToDebug
114     Value = H.
115
116
117 %used TODO: optimistisches Bergsteigen
118 %in Suche_Modul_Allgemein.pl gelöst!
119
120 %used TODO: Bergsteigen mit Backtracking
121 %in Suche_Modul_Allgemein.pl gelöst!
122
123 %</Suchverfahren>
124 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
125 %<Heuristiken>
126 % Anzahl der Elemente des Ziels, die noch an der falschen Position
127 sind.
128 /* heuristik(State,Value) :-
129     goal_description(Ziel),
130     differenzmenge(Ziel,State,Differenz),
131     length(Differenz,Value).
132 */
133
134 % Anzahl der Elemente, die schon an der richtigen Position sind
135 heuristik(State,Value) :-
136     goal_description(Ziel),
137     schnittmenge(State,Ziel,Schnitt),
138     length(Schnitt,AnzSchnitt),
139     length(Ziel,AnzZiel),

```

```

70
71 action(pick_up(X),
72     [handempty, clear(X), on(table,X)],
73     [handempty, clear(X), on(table,X)],
74     [holding(X)]).
75
76 action(pick_up(X),
77     [handempty, clear(X), on(Y,X), block(Y)],
78     [handempty, clear(X), on(Y,X)],
79     [holding(X), clear(Y)]).
80
81 action(put_on_table(X),
82     [holding(X)],
83     [holding(X)],
84     [handempty, clear(X), on(table,X)]).
85
86 action(put_on(Y,X),
87     [holding(X), clear(Y)],
88     [holding(X), clear(Y)],
89     [handempty, clear(X), on(Y,X)]).
90
91
92 % Hilfskonstrukt, weil das PROLOG "subset" nicht die Unifikation
  von Listenelementen
93 % durchföhrt, wenn Variablen enthalten sind. "member" unifiziert

```

```

139     Value is (AnzZiel - AnzSchnitt).
140
141
142 %</Heuristiken>
143
144
145 % Mengenoperationen
146 %-----
147 % differenzmenge(Menge_A, Menge_B, Differenzmenge)    "A ohne B"
148 differenzmenge([], _, []).
149 differenzmenge([Element|Tail], Menge, Rest) :-
150     member(Element, Menge), !,
151     differenzmenge(Tail, Menge, Rest).
152 differenzmenge([Head|Tail], Menge, [Head|Rest]) :-
153     differenzmenge(Tail, Menge, Rest).
154
155 % schnittmenge(Menge_A, Menge_B, Menge_C)
156 schnittmenge([], _, []).
157 schnittmenge([Element|Tail], Liste, Schnitt) :-
158     member(Element, Liste), !,
159     Schnitt = [Element|Rest],
160     schnittmenge(Tail, Liste, Rest).
161 schnittmenge(_|Tail, Liste, Rest) :-
162     schnittmenge(Tail, Liste, Rest).
163 %</TEIL 2>
164
165 action(pick_up(X),
166     [handempty, clear(X), on(table,X)],
167     [handempty, clear(X), on(table,X)],
168     [holding(X)]).
169
170 action(pick_up(X),
171     [handempty, clear(X), on(Y,X), block(Y)],
172     [handempty, clear(X), on(Y,X)],
173     [holding(X), clear(Y)]).
174
175 action(put_on_table(X),
176     [holding(X)],
177     [holding(X)],
178     [handempty, clear(X), on(table,X)]).
179
180 action(put_on(Y,X),
181     [holding(X), clear(Y)],
182     [holding(X), clear(Y)],
183     [handempty, clear(X), on(Y,X)]).
184
185
186 % Hilfskonstrukt, weil das PROLOG "subset" nicht die Unifikation von
  Listenelementen
187 % durchföhrt, wenn Variablen enthalten sind. "member" unifiziert hingegen.

```

```

hingegen.
94 %
95 mysubset([],_).
96 mysubset([H|T],List):-
97     member(H,List),
98     mysubset(T,List).
99
100
101 expand_help(State,Name,NewState):-
102     "Action suchen"
103     "Conditions testen"
104     "Del-List umsetzen"
105     "Add-List umsetzen".
106
107 expand(_,State,_),Result):-
108     findall((Name,NewState,_),expand_help(State,Name,NewState),Result).

```

```

188 %
189 mysubset([],_).
190 mysubset([H|T],List):-
191     member(H,List),
192     mysubset(T,List).
193
194
195 expand_help(State,Name,NewState):-
196     %used TODO:"Action suchen"
197     action(Name, ConditionList, DeleteList, AddList),
198
199     %used TODO:"Conditions testen" Folie: "Unter welchen Vorbedingungen ist
    die Aktion anwendbar?"
200     mysubset(ConditionList,State),
201
202     %used TODO:"Del-List umsetzen" Folie: "Welche Formeln gelten nach
    Ausführung der Aktion nicht mehr?"
203     subtract(State, DeleteList, TempNewState),
204     /*writeln(['STATE: ',State]), %%DEBUG
205     writeln(['DEL: ', DeleteList]),
206     writeln(['AFTER: ',TempNewState]),nl,*/
207
208     %used TODO:"Add-List umsetzen". Folie: "Welche Formeln kommen nach
    Ausführung der Aktion hinzu?"
209     append(TempNewState,AddList,NewState).
210     /*writeln(['ADD: ', AddList]), %%DEBUG
211     writeln(['AFTER: ',NewState]),nl.*/
212
213
214 expand(_,State,_),Result):-
215     findall((Name,NewState,_),expand_help(State,Name,NewState),Result).

```