Fakultät Informatik/Mathematik
Lehrstuhl für Künstliche Intelligenz / Theoretische Informatik

HOCHSCHULE FÜR
TECHNIK UND WIRTSCHAFT
DRESDEN
UNIVERSITY OF APPLIED SCIENCES

Hausaufgabe

KÜNSTLICHE INTELLIGENZ

AUFGABE 16

Falk-Jonatan Strube
Bibliotheksnummer: s74053

Praktikum von
Prof. Dr. Boris Hollas

11. Mai 2017

```
command(CMD, POS_A, POS_B) :- moveCommand(CMD, CLR, DIR), positions(POS_A, CLR1_1, POS1_1, CLR2_1, POS2_1), move(CLR, DIR, CLR1_1, POS1_1, CLR2_1, POS2_1, POS_B).

% Prüfe/Extrahiere Move-Befehl: moveCommand(Liste von Strings, Farbe, Richtung).
moveCommand([move|LST], CLR, DIR) :- color(LST, CLR, [DIR|_]).
% Prüfe/Extrahiere Position: position(Liste von Strings, Farbe 1, ist an Position 1, Farbe 2, ist an Position 2).
positions(LST, CLR1, POS1, CLR2, POS2) :- color(LST, CLR1, R1), coordinate(R1, POS1, R2), color(R2, CLR2, R3), coordinate(R3, POS2, []).
% Extrahiere Koordinate: coordinate(Liste von Strings, Koordinate, Rest).
coordinate([C|R], C, R).
% Finde/Extrahiere Farbe und eliminiere Unwichtiges danach: color(Liste von Strings, Farbe der Stringkette, Rest).
color(LST, CLR, R) :- ( match(black, LST, R1, CLR) ; match(white, LST, R1, CLR) ), stuff(R1, R) .
% Eliminiere unwichtiges: stuff(Liste von Strings, Rest).
stuff(LST, R) :- ( match(rook, LST, R1, _ ); match(is, LST, R1, _ ) ; match(at, LST, R1, _ ) ), stuff(R1, R).
stuff(LST, LST).
% Bewege, falls möglich die richtige Figur: move(zu bewegende Farbe, Richtung, Farbe 1, Position 1, Farbe 2, Position 2).
% zuerst normieren:
move(CLR, DIR, white, POS1_1, black, POS2_1, POS_B) :- move(CLR, DIR, black, POS2_1, white, POS1_1, POS_B).
% dann bewegen:
move(black, DIR, black, POSB, white, POSW, POS_B) :- changePos(POSB, DIR, POSB2), validPos(POSB2, POSB, POSW, POSBF), POS_B = [black, at, POSBF, white, at, POSW].
move(white, DIR, black, POSB, white, POSW, POS_B) :- changePos(POSW, DIR, POSW2), validPos(POSW2, POSW, POSB, POSWF), POS_B = [black, at, POSB, white, at, POSWF].
% Gib veränderte Position aus: changePos(Originalposition, Richtung, Position nach Bewegung).
changePos(POSA, forward, POSB) :- xCoord(POSA, X), yCoord(POSA, Y), YB is (Y+1), POSB = (X, YB).
changePos(POSA, back, POSB) :- xCoord(POSA, X), yCoord(POSA, Y), YB is (Y-1), POSB = (X, YB).
changePos(POSA, left, POSB) :- xCoord(POSA, X), yCoord(POSA, Y), XB is (X-1), POSB = (XB, Y).
changePos(POSA, right, POSB) :- xCoord(POSA, X), yCoord(POSA, Y), XB is (X+1), POSB = (XB, Y).
% Gebe korrekte Position aus (verrückte, wenn Prüfung ok, sonst Originalposition): validPos(zu prüfende Position, Originalposition, Position der anderen Figur, Rückgabeposition).
validPos(POSA, _, POSB, POSA) :- onPlan(POSA), POSA \== POSB.
validPos(_, POSAo, _, POSAo).
% Prüfe ob Position auf dem Plan ist:
```


```
29 onPlan(POS) :- xCoord(POS, X), X < 9, X > 0, yCoord(POS, Y), Y < 9,
       Y > 0.
30 % Gebe Position von Argumenten aus:
31 xCoord(POS, X) :- arg(1, POS, X).
32 yCoord(POS, Y) :- arg(2, POS, Y).
33
34 % Die gute, alte Match-Funktion
35 match(X, [X|Rest], Rest, X).
36
37 % Testeingaben und Ergebnisse:
38 %?- command([move , black , rook , forward], [black , is, at, (1,1)
       , white , rook , at, (8,8)], R).
39 % R = [black , at, (1, 2), white , at, (8, 8)] .
40 %?- command([move , white , left], [white , at, (5,5), black , rook
       , is, at, (2,2)], R).
41 % R = [black , at, (2, 2), white , at, (4, 5)] .
42 %?- command([move , white , left], [white , at, (5,5), black , rook
       , is, at, (4,5)], R).
43 % R = [black , at, (4, 5), white , at, (5, 5)] .
```