

```
1 /*
2  * -----
3  * Logische Programmierung, ueb 6.1-6.2, 16.12.2002
4  * (c)Marc Dörflinger, Roger Nösberger, Philip Iezzi
5  * -----
6  */
7
8 % fsa wird mit der liste aus den wörtern des zu parsenden satzes aufgerufen
9
10 fsa(List) :-
11     initial(A),
12     fsa(List, A).
13
14 % wenn die liste leer ist und State dem endzustand entspricht, dann ist fertig
15
16 fsa([], State) :-
17     final(State).
18
19
20 % wenn es einen übergang von State nach NewState mit Element gibt, dann wird
21 die fsa mit
22 % dem NewState und dem rest der Liste aufgerufen.
23
24 fsa([Element|List], State) :-
25     transition(State, Element, NewState),
26     fsa(List, NewState).
27
28 % Folge von Symbolen:
29 % [the, cat, is, on, the, mat, '.']
30 % [the, dog, is, on, the, mat, '.']
31
32 initial(z0).
33 final(e).
34 transition(z0, the, z1).
35 transition(z1, cat, z2).
36 transition(z1, dog, z2). % nicht-deterministisch!
37 transition(z2, is, z3).
38 transition(z3, on, z4).
39 transition(z4, the, z5).
40 transition(z5, mat, z6).
41 transition(z6, '.', e).
42
43
44
45
46 /*
47  * -----
48  * Logische Programmierung, ueb 6.3, 16.12.2002
49  * -----
50  */
51
52 initial([the, cat, is, on, the, mat, '.']).
53 initial([the, dog, is, on, the, mat, '.']).
54
55 fsa(List) :-
56     initial(StartState),
57     accept(List, StartState).
58
59 accept([], []).
60
61 accept([H|Tail1], [H|Tail2]) :-
62     accept(Tail1, Tail2).
63
64
65
66
67
```

```
68
69 /*
70  * -----
71  * Logische Programmierung, ueb 6.4, 16.12.2002
72  * -----
73  */
74
75 fsa(List) :-
76     states(T),
77     fsa(List, T).
78
79 fsa([], []).
80
81 fsa([Element|List], [CurrentState|OtherStates]) :-
82     T =.. [CurrentState, Element],
83     T,
84     fsa(List, OtherStates).
85
86 % Grammatik
87 states([article, noun, verb, preposition, article, noun, punctuation]).
88
89 % Syntax
90 article(the).
91 article(a).
92 noun(cat).
93 noun(dog).
94 noun(mat).
95 verb(is).
96 preposition(on).
97 punctuation(' ').
```