Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

Logik

Sommersemester 2023
9. Aufgabenblatt

Aufgabe 15 (7 Punkte)

Schreiben Sie folgende Prädikate in Prolog. Sie können dabei das im System vorhandene Prädikat append/3 mit dem die zweite übergebene Liste an die erste angehängt und zum dritten Parameter wird (append([a,b], [c,d], [a,b,c,d])), nutzen. Natürlich können Sie sich auch weitere Hilfsprädikate schreiben. Achten Sie darauf, dass die hier gezeigten und von der Veranstaltungsseite erhältlichen Testprädikate aus AufgabenListen.pl als Ergebnis true liefern. Am Ende soll testAll true als Ergebnis haben. Beachten Sie, dass nicht nur die Tests laufen müssen sondern Ihre Prädikate auch die angegebenen Lösungen in der Konsole berechnen können sollen.

a) Schreiben Sie ein Prädikat anzahl/3, das für eine gegebene Liste und ein Element berechnet, wie oft dieses Element in der Liste vorkommt.

```
?- anzahl([y, o, l, o], o, Y).
Y = 2 .

testAnzahl :-
  anzahl([], _, 0),
  anzahl([a], b, 0),
  anzahl([a], a, 1),
  anzahl([a, b, 42, a], a, 2).
```

b) Schreiben Sie ein Prädikat einmalig/2, das für eine gegebene Liste eine Liste berechnet, die alle Elemente der ersten Liste nur einmal enthält und dabei die Reihenfolge der ersten Vorkommen einhält.

```
?- einmalig([a, b, b, a], X).
X = [a, b] .

testEinmalig :-
  einmalig([], []),
  einmalig([a], [a]),
  einmalig([a,b], [a,b]),
  einmalig([a,b,a], [a,b]),
  einmalig([a,b,b,b,a,a], [a,b]).
```

c) Schreiben Sie ein Prädikat mymerge/3, das für zwei gegebene Listen eine dritte Liste berechnet, die abwechselnd, beginnend mit der ersten Liste Elemente der beiden Listen enthält. Ist eine Liste abgearbeitet, wird der Rest der anderen Liste angehängt.

```
?- mymerge([k, u, e, n],[a, f], X).
[k,a]
[u,f]
X = [k, a, u, f, e, n] .

testMerge :-
  mymerge([], [], []),
  mymerge([a], [b], [a, b]),
  mymerge([a, c], [b], [a, b, c]),
  mymerge([a, c], [b, d, e, f], [a, b, c, d, e, f]),
  mymerge([a, c, x, y], [b, d], [a, b, c, d, x, y]).
```

d) Schreiben Sie ein Prädikat findePaar/2, das für eine gegebene Liste und eine zweielementige Liste prüft, ob diese zweielementige Liste in genau dieser Reihenfolge innerhalb der ersten übergebenen Liste vorkommt.

```
?- findePaar([b,a,b,a,b,a,b,a,b], [a,b]).
true .

testFindePaar :-
  findePaar([a,b], [a,b]),
  findePaar([a,b,c,d], [a,b]),
  findePaar([a,a,a,b], [a,b]),
  findePaar([a,a,a,b,b,b], [a,b]) ,
  not(findePaar([b,b,b,a,a,a], [a,b])),
  not(findePaar([a,c,b,a,c,b], [a,b])).
```

e) Schreiben Sie ein Prädikat dreiAlsSumme/2, das für eine gegebene Liste von Zahlen und eine konkrete Zahl überprüft, ob die konkrete Zahl als Summe dreier beliebiger unterschiedlicher Elemente der Liste berechnet werden kann.

```
?- dreiAlsSumme([1,2,4,8], X).
X = 7 ;
X = 14 ;
X = 13 ;
X = 11 ;
false.

testDreiAlsSumme:-
  not(dreiAlsSumme([], 0)),
  not(dreiAlsSumme([1], 1)),
  not(dreiAlsSumme([1,1], 2)),
  not(dreiAlsSumme([2,4,6], 11)),
  not(dreiAlsSumme([2,44,4,6,4,2], 11)),
  dreiAlsSumme([2,2,2], 6),
  dreiAlsSumme([3,2,4,5], 9),
  dreiAlsSumme([3,2,4,5], 10),
  dreiAlsSumme([3,2,4,5], 11),
  dreiAlsSumme([3,2,4,5], 12),
  dreiAlsSumme([2,17,14,3,14,17,4], 9).
```

f) (optional) Schreiben Sie ein Prädikat split/4, das für eine gegebene Liste die möglichen Zerlegungen in den Anfang der Liste, das dann folgende einzelne Element der Liste und den Rest der Liste berechnet. Es gibt damit für eine Liste mit n Elementen insgesamt n Lösungen, die z. B. durch die Eingabe eines Semikolons nacheinander angezeigt werden.

```
?- split([a,b,c], L, E, R).
L = [],
E = a,
R = [b, c] ;
L = [a],
E = b,
R = [c] ;
```

```
L = [a, b],
E = c,
R = [] ;
false.

testSplit:-
  not(split([], _, _, _)),
  split([a], [], a, []),
  split([a,b,c,d], [], a, [b,c,d]),
  split([a,b,c,d], [a], b, [c,d]),
  split([a,b,c,d], [a, b], c, [d]),
  split([a,b,c,d], [a,b,c], d, []).
```

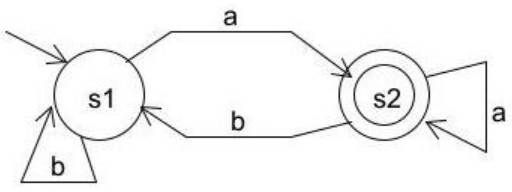
g) (optional) Schreiben Sie ein Prädikat anHaeufigsten/3, das für eine gegebene Liste, ein Element berechnet, dass am häufigsten in der Liste vorkommt und angibt, wie oft dieses Element vorkommt. Da es mehrere Elemente geben kann, die am Häufigsten vorkommen, sollen alle Lösungen nacheinander mit einem Semikolon ausgegeben werden.

```
?- amHaeufigsten([a,b,b,a], B, C).
B = b,
C = 2 ;
B = a,
C = 2 ;
false.

testAmHaeufigsten :-
  not(amHaeufigsten([], _, _)),
  amHaeufigsten([a], a, 1),
  amHaeufigsten([a,a], a, 2),
  amHaeufigsten([b,a,a], a, 2),
  amHaeufigsten([a,b,a], a, 2),
  amHaeufigsten([a,a,b], a, 2),
  amHaeufigsten([b,a,a,b], a, 2),
  amHaeufigsten([b,a,a,b], b, 2),
  amHaeufigsten([b,a,a,b,c], a, 2),
  amHaeufigsten([b,a,a,b,c], b, 2),
  amHaeufigsten([b,a,a,b,c,c], a, 2),
  amHaeufigsten([b,a,a,b,c,c], b, 2),
  amHaeufigsten([b,a,a,b,c,c], c, 2).
```

Aufgabe 16 (2 Punkte)

Ein endlicher deterministischer Automat dient zur Berechnung, ob eine Folge von Zeichen (hier eine Liste von Konstanten) akzeptiert wird oder nicht. Zur Berechnung hat der Automat einen Startzustand (mit eingehendem Pfeil gekennzeichnet) und eine Überführungsfunktion mit der festgelegt wird, wie aus einem aktuellen Zustand mit einem eingelesenen Zeichen in



einen Folgezustand übergegangen werden kann. Die Überführungsfunktion ist graphisch als gerichtete Kante zwischen aktuellem und Folgezustand, markiert mit dem gelesenen Zeichen. Eine Teilmenge der endlichen Menge von Zuständen sind Endzustände, die mit einem Doppelkreis markiert sind. Ist die Eingabe vollständig abgearbeitet und der Automat befindet sich in einem der Endzustände, wird die Eingabe akzeptiert, sonst nicht. Der Automat auf der rechten Seite akzeptiert z. B. ababaa, aber nicht abb.
Überlegen Sie sich, wie generell ein konkreter gegebener Automat mit Prolog umgesetzt werden kann, dabei ist u. a. die Überführungsfunktion umzusetzen.
Setzen Sie dann den Beispielautomaten mit um, am Ende sollte folgender Programmablauf möglich sein.

```
?- akzeptiert([]).
false.

?- akzeptiert([a]).
true.

?- akzeptiert([a,b,a,b,a,a]).
true.

?- akzeptiert([a,b,b]).
false.
```