Logik Prof. Dr. H. Peter Gumm
Wintersemester 22/23

Übungszettel 07

Abgabetermin: 10. Dezember 2021, 18:00 Uhr. Die Abgabe erfolgt über Ihr Ilias-Tutorium.

Aufgabe 1 (Arithmetik)**.** Programmieren (und testen) Sie in Prolog

1. `find(X,L,N)` liefert die erste Position der Liste `L`, an der das Element `X` vorkommt. Falls `X` nicht in `L` vorkommt, sei `N=0`.
   `find(3,[1,4,5,3,2,3],N).` ⇝ `yes. N / 4` und `find(6,[1,4,3,5,2,3],N).` ⇝ `yes. N / 0`
2. `sumAll`, berechnet die Summe aller Zahlen aus einer Liste `L` :
   `sumAll([1,3,-7],R).` ⇝ `yes. R / -3`
3. `mapSquare` quadriert alle Elemente in einer Liste:
   `mapSquare([1,3,-7],R).` ⇝ `yes. R / [1,9,49]`
4. `filterPos` filtert alle Elemente aus einer Liste von Zahlen, die positiv sind:
   `filterPos([1,-2,-3,3,-7],R).` ⇝ `yes. R / [1,3]`

(4 Punkte)

Aufgabe 2 (Listenoperationen)**.** Implementieren Sie in Prolog die folgenden Listenoperationen.

(a) `ersetze(X,Y,L,R)`: `R` entsteht aus `L` durch Ersetzen *jedes* Vorkommens von `X` durch `Y`:
`ersetze(3,5,[1,5,7,3,9,3,3],R).` ⇝ `yes. R / [1,5,7,5,9,5,5]`

(b) `remDups(L,R)` liefert in `R` die Liste aller Elemente aus L, ohne Duplikate:
`remDups([1,5,1,7,3,9,3,3],R).` ⇝ `yes. R / [5,1,7,9,3]`

(c) `vereinigung`, liefert die (mengentheoretische) Vereinigung zweier Listen. Sie dürfen voraussetzen, dass die beiden Listen duplikatfrei sind. Das Ergebnis soll auch kein Duplikat enthalten:
`vereinigung([1,3,5,7],[7,4,9,3],R).` ⇝ `yes. R / [1,5,7,4,9,3]`

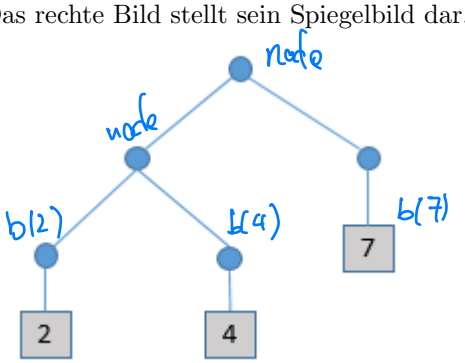
(d) `diff` lieferte die (mengentheoretische) Differenz zweier Listen, also `diff(Xs,Ys,R)` liefert in `R` die Elemente der Liste `Xs`, die nicht auch in `Ys` vorkommen.
`diff([1,3,5,7],[7,4,9,3],R).` ⇝ `yes. R / [1,5]`

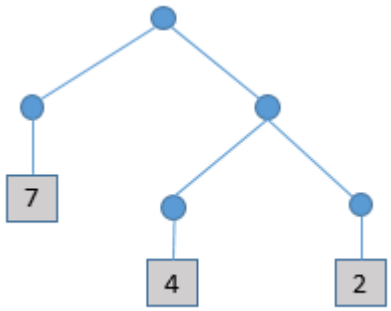
(4 Punkte)

Aufgabe 3 (Arithmetik)**.** Ein *bTree* soll einen Binärbaum repräsentieren, der seine Daten nur Blättern, trägt:

- *blatt(X)* sei der *bTree*, der nur aus einem Blatt mit Inhalt X besteht
- *node(left,right)* sei der *bTree* mit linkem Teilbaum *left* und rechtem Teilbaum *right*.

Beispiel: Der Term $node(node(blatt(2), blatt(4)), blatt(7))$ repräsentiert den Binärbaum der linken Figur. Das rechte Bild stellt sein Spiegelbild dar.





(a) Schreiben Sie ein Prolog-Programm *mirror*, das zu einem Binärbaum sein Spiegelbild liefert.

(b) Schreiben Sie ein Prolog-Programm, das die Daten eines Binärbaums in *inorder*-Reihenfolge einer Liste speichert. Für den linken Binärbaum wäre das Ergebnis die Liste $[2, 4, 7]$.

(2+2 Punkte)

Aufgabe 4 (Arithmetik)**.** Die Programme `sat.pl`, `dpll.pl` und `resolve.pl`, die in der Vorlesung diskutiert wurden, finden Sie im *ilias*. In dieser Aufgabe sollen Sie diese Programme diskutieren und modifizieren.

1. Modifizieren Sie das Programm `sat.pl` so dass nicht nur die erste der gefundenen Belegungen ausgegeben wird, sondern alle.
2. Modifizieren Sie nun `sat.pl`, so dass die Liste aller erfüllenden Belegungen erzeugt wird.
3. Warum ist `delmem` in `dpll.pl` so kompliziert? Testen Sie, ob `dpll` auch funktionieren würde, wenn man `delmem` ersetzt durch:
```
delmem(A,[A|As],Bs)      :- delmem(A,As,Bs).
delmem(A,[B|Bs],[B|Rs]) :- delmem(A,Bs,Rs).
```
4. Was würde sich in `resolve.pl` ändern, wenn man `pickTwo` ersetzt durch:
```
pickTwo(Xs,A,B) :- member(A,Xs),member(B,Xs), A \= B.
```

(4 Punkte)

```
[[1, -3, 7], [-1, -2], [3, -7]] :-
    sat2([[-1, -2], [3, -7]], [1], Y)
    sat2
```

[1, -3, 7], [-1, -2], [3, -7] [3, -2, 1]

[-1, -2], [3, -7]

[-2], [3, -7] [-2, 1]

[3, -7] [3, -2, -1] ✓

[7, -2, 3]

1 -1 7 +

[[1, -3, 7], [-1, -2], [3, -7]], [...], [3, -2, -1]

[[]...[]..]

zugehörige Belegung