Prof. Dr. J. Giesl
D. Cloerkes, S. Dollase, N. Lommen, D. Meier, F. Meyer

Tutoraufgabe 1 (Überblickswissen):

a) Listen werden in Prolog intern mit der leeren Liste `[]` und dem Listenkonstruktor `.` repräsentiert. Inwiefern ist die Notation mit dem Listenkonstruktor `cons` und der leeren Liste `nil` dazu analog? Weil diese beiden Schreibweisen nicht sonderlich praktisch sind, schreibt man Listen häufig in der Schreibweise `[Kopf|Rest]`. Wie hängt diese Listenschreibweise mit den anderen zusammen?

b) Sowohl Prädikate als auch Funktionen haben in Prolog Bezeichner, die mit einem Kleinbuchstaben beginnen. Obwohl sie also leicht zu verwechseln sind, haben sie ganz unterschiedliche Bedeutungen. Wozu dienen Funktionen? Was sind im Gegensatz dazu Prädikate?

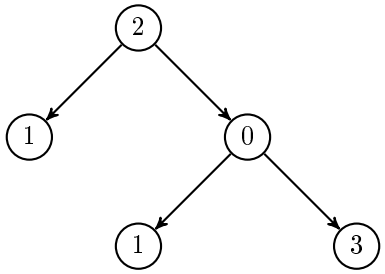
c) Was ist der Unterschied zwischen Unifikation und Pattern Matching?

d) Spielt es eine Rolle, in welcher Reihenfolge die Klauseln eines Prolog-Programms stehen?

e) Was sind Anwendungsgebiete der Logikprogrammierung?

Tutoraufgabe 2 (BinTree):

Natürliche Zahlen lassen sich in Prolog (oder anderen deklarativen Sprachen) durch die Peano-Notation als Terme darstellen. Dabei stellt die Konstante `0` die Zahl 0 dar und für eine Zahl n dargestellt durch den Term `N` stellt `s(N)` die Zahl $n+1$ dar. So wird z. B. die Zahl 3 durch den Term `s(s(s(0)))` dargestellt.
Binäre Bäume können in Prolog folgendermaßen als Terme dargestellt werden. Sei n eine natürliche Zahl dargestellt durch den Term `N`. Dann repräsentiert der Term `leaf(N)` einen Baum mit nur einem Blatt, welches den Wert n enthält. Für zwei Bäume x und y dargestellt durch die Terme `X` und `Y` repräsentiert der Term `node(X,N,Y)` einen binären Baum mit einem Wurzelknoten, der den Wert n enthält und die Teilbäume x und y hat. Als Beispiel ist nachfolgend ein binärer Baum und seine Darstellung als Term angegeben.




```
node(leaf(s(0)),s(s(0)),node(leaf(s(0)),0,leaf(s(s(s(0))))))
```

a) Schreiben Sie ein Prädikat `increment/2` in Prolog, wobei `increment(B,IncB)` genau dann wahr sein soll, wenn sich der Baum `IncB` aus dem Baum `B` ergibt, indem jede Zahl, die in einem inneren Knoten oder Blatt steht, um eins erhöht wird. Beispielsweise soll der Aufruf

```
increment(node(leaf(s(0)),s(s(0)),leaf(0)),Res)
```

das Ergebnis `Res = node(leaf(s(s(0))),s(s(s(0))),leaf(s(0)))` liefern.

b) Schreiben Sie ein Prädikat `append(XS,YS,Res)`, das die Listen `XS` und `YS` hintereinanderhängt. Ein Aufruf von `append([a,b,c],[d,e],Res)` würde das Ergebnis `Res = [a,b,c,d,e]` liefern.

c) Es gibt verschiedene Verfahren, um einen Baum systematisch zu durchsuchen. Man unterscheidet zwischen Pre-, In- und Postorder-Traversierung. Bei einer Preorder-Traversierung wird zuerst die Wurzel (W) eines Baums durchsucht, dann der linke Teilbaum (L) und anschließend der rechte Teilbaum (R). Bei Inorder ist die Reihenfolge LWR und bei Postorder LRW. Für den Beispielbaum aus der Abbildung ergeben sich folgende Ausgaben:

- Preorder: 2, 1, 0, 1, 3
- Postorder: 1, 1, 3, 0, 2
- Inorder: 1, 2, 1, 0, 3

Sie sollen nun ein Prädikat `inorder(B,Res)` schreiben, das alle Knoten eines Baumes `B` in **In**order-Reihenfolge in die Liste `Res` einträgt. Wenn `B` der Baum aus der Abbildung ist, so würde `inorder(B,Res)` das Ergebnis `Res = [s(0),s(s(0)),s(0),0,s(s(s(0)))]` liefern.

Hinweise:

- Sie dürfen die Funktion `append` aus Teilaufgabe b) verwenden.

Tutoraufgabe 4 (Unifikation):

In dieser Aufgabe sollen allgemeinste Unifikatoren bestimmt werden. Sie sollten diese Aufgabe ohne Hilfe eines Rechners lösen, da Sie zur Lösung von Aufgaben dieses Typs auch in der Klausur keinen Rechner zur Verfügung haben.
Nutzen Sie den Algorithmus zur Berechnung des allgemeinsten Unifikators (MGU) aus der Vorlesung, um die folgenden Termpaare auf Unifizierbarkeit zu testen.
Geben Sie neben dem Endergebnis σ auch die Unifikatoren $\sigma_1, \sigma_2, \ldots, \sigma_n$ für die direkten Teilterme der beiden Terme an. Sollte ein σ_i nicht existieren, so begründen Sie kurz, warum die Unifikation fehlschlägt. Geben Sie in diesem Fall an, ob es sich um einen *clash failure* oder einen *occur failure* handelt.

(i) **f(X,X,g(X,X))** und **f(Y,a,g(b,Y))**

(ii) **f(X,g(a,Y))** und **f(h(Y),g(Z,h(Z)))**

(iii) **f(g(X,a),X)** und **f(g(Y,X),g(Y,X))**

(iv) **f(W,g(Y),Y,W)** und **f(g(X),X,g(Z),Z)**

(v) **f(g(X,Z),Z,g(Z,X))** und **f(Y,a,g(Z,g(W,X)))**

Beispiel:
Für **f(A, g(c), h(Y, Y))** und **f(c, X, h(A, X))** ist folgende Lösung anzugeben:

$\sigma_1 = \{A = c\}$

$\sigma_2 = \{X = g(c)\}$

$\sigma_3 = \text{mgu}(h(Y,Y), h(c, g(c)))$ existiert nicht, da c und $g(c)$ nicht mit dem gleichen Symbol beginnen. Folglich liegt ein *clash failure* vor.

Für **f(A, g(c), h(Y, g(c)))** und **f(c, X, h(A, X))** ist folgende Lösung anzugeben:

$\sigma_1 = \{A = c\}$

$\sigma_2 = \{X = g(c)\}$

$\sigma_3 = \{Y = c\}$

$\sigma = \sigma_3 \circ \sigma_2 \circ \sigma_1 = \{A = c, X = g(c), Y = c\}$

Tutoraufgabe 6 (Beweisbäume):

Betrachten Sie die Anfrage `?- t(c,Z).` auf folgendem Prolog-Programm:

```
t(X,c) :- t(X,b).
t(X,X) :- p(X,a).
t(X,b) :- t(c,X).
t(c,b).
```

a) Geben Sie den zugehörigen Beweisbaum (SLD-Baum) bis einschließlich Höhe 3 an. Die Höhe eines Baums ist der längste Pfad von der Wurzel bis zu einem Blatt. Ein Baum, welcher nur aus einem Blatt besteht, hat also die Höhe 0. Markieren Sie unendliche Pfade mit ∞ und Fehlschläge mit $(fail)$. Geben Sie alle Antwortsubstitutionen zur obigen Anfrage an und geben Sie an, welche dieser Antwortsubstitutionen von Prolog gefunden werden.

b) Strukturieren Sie das gegebene Programm in ein logisch äquivalentes Programm um, sodass Prolog mit seiner Auswertungsstrategie **mindestens zwei** Lösungen zur gegebenen Anfrage findet. Der Beweisbaum (SLD-Baum) muss nicht endlich sein! Bei dieser Umstrukturierung dürfen Sie nur die Reihenfolge der Prolog-Klauseln verändern.

Tutoraufgabe 8 (Arithmetik mit Prolog):

Formulieren Sie ein Prolog-Programm mit einem Prädikat `squares(N, R)`, das wahr ist, wenn `N` ≥ 1 gilt und `R` die absteigende Liste der Quadratzahlen von `N`^{2} bis 1 ist. Beispielsweise soll `squares(5, [25, 16, 9, 4, 1])` wahr sein. Verwenden Sie dafür die Gleichung $k^2 = (k-1)^2 + 2*(k-1) + 1$ und nicht direktes Quadrieren. Benutzen Sie das vordefinierte Prädikat `is/2`.