Prof. Dr. M. Müller C. Aschermann, J. Hensel, J. Protze, P. Reble

Allgemeine Hinweise:

- Die **Hausaufgaben** sollen in Gruppen von je **3 Studierenden** aus der **gleichen Kleingruppenübung (Tutorium)** bearbeitet werden. **Namen und Matrikelnummern** der Studierenden sind auf jedes Blatt der Abgabe zu schreiben. **Heften bzw. tackern Sie die Blätter!**
- Die **Nummer der Übungsgruppe** muss **links oben** auf das **erste Blatt** der Abgabe geschrieben werden. Notieren Sie die Gruppennummer gut sichtbar, damit wir besser sortieren können.
- Die Lösungen müssen **bis Dienstag, den 09.02.2016 um 15:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Alternativ können Sie die Lösungen auch vor der Abgabefrist direkt bei Ihrer Tutorin/Ihrem Tutor abgeben.
- In einigen Aufgaben müssen Sie in Prolog programmieren und `.pl`-Dateien anlegen. **Drucken** Sie diese aus **und** schicken Sie sie per **E-Mail** vor Dienstag, dem 09.02.2016 um 15:00 Uhr an Ihre Tutorin/Ihren Tutor.
  Stellen Sie sicher, dass Ihr Programm von `SWI-Prolog` **akzeptiert** wird, ansonsten werden keine Punkte vergeben.

Tutoraufgabe 1 (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(h(X),g(Y),X,Y)** und **f(Z,g(Z),a,Z)**

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

(iii) **f(X,h(Y),Y)** und **f(g(Z),X,Z)**

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

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

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

σ_3 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:

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

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

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

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

Aufgabe 2 (Unifikation): (5 Punkte)

In dieser Aufgabe sollen allgemeinste Unifikatoren bestimmt werden. Sie sollten iese 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(g(X,Y),g(Y,Z),g(Z,X))** und **f(g(A,a),g(B,B),g(c,C))**

(ii) **g(X,Y,Y,X)** und **g(a,s(X),s(Z),Z)**

(iii) **f(X,Z,Z)** und **f(Y,Y,s(X))**

(iv) **f(X,s(Y),X,Y)** und **f(Z,Z,s(b),a)**

(v) **f(X,s(s(Z)),Z)** und **f(s(Y),X,a))**

Tutoraufgabe 3 (Beweisbäume):

Betrachten Sie die Anfrage `?- q(Z,s(0)).` auf folgendem Prolog-Programm:

```
q(X, Y) :- q(s(X), Y).
q(0, Y) :- h(s(Y), Y).
q(s(X), X).
h(X, X) :- q(s(X), X).
```

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

b) Strukturieren Sie das gegebene Programm so in ein logisch äquivalentes Programm um, dass Prolog mit seiner Auswertungsstrategie **mindestens eine** Lösung zur gegebenen Anfrage findet. Der Beweisbaum (SLD-Baum) muss nicht endlich sein! Sie brauchen den SLD Baum nicht angeben.

Hinweis: Bei dieser Umstrukturierung dürfen Sie nur die Reihenfolge der Prolog-Klauseln verändern.

Aufgabe 4 (Beweisbäume): (5 + 1 = 6 Punkte)

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

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

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

b) Strukturieren Sie das gegebene Programm so in ein logisch äquivalentes Programm um, dass Prolog mit seiner Auswertungsstrategie **mindestens eine** Lösung zur gegebenen Anfrage findet. Der Beweisbaum (SLD-Baum) muss nicht endlich sein!

Hinweis: Bei dieser Umstrukturierung dürfen Sie nur die Reihenfolge der Prolog-Klauseln verändern.

Tutoraufgabe 5 (Geschenksocken):

In dieser Aufgabe betrachten wir eine eigene Datenstruktur für Geschenksocken, sodass man diese am Kamin aufhängen kann, um Geschenke zu sammeln. Jede solche Socke hat Platz für eine feste Anzahl von Geschenken. Hierbei hat jedes Geschenk ein Gewicht und einen Namen. Eine Socke wird durch eine Liste von Einträgen repräsentiert, wobei jeder Eintrag entweder `luft` oder `geschenk(..., ...)` ist. Ein Geschenk `geschenk(gewicht, name)` speichert das Gewicht als Zahl in `gewicht`, den Namen als String in `name`.

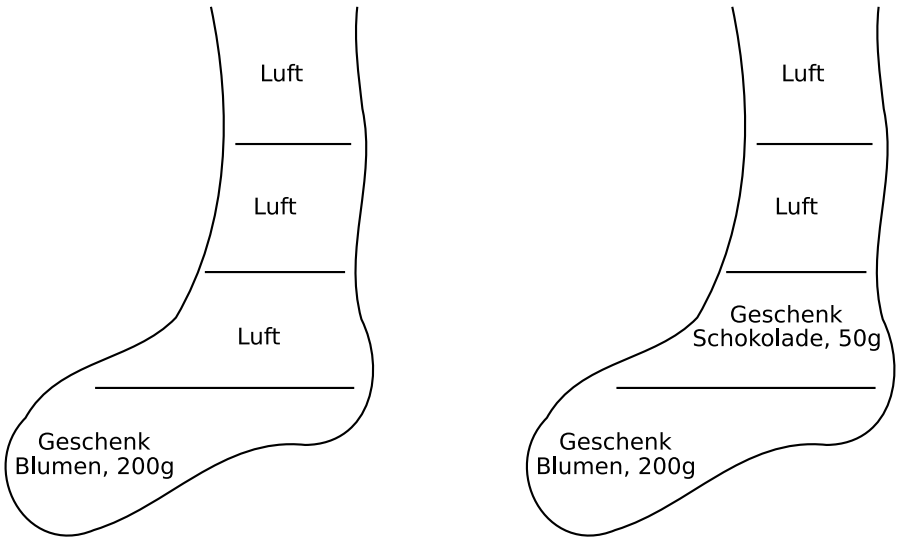


Abbildung 1: Geschenksocken `S` (links) und `T` (rechts)

a) Implementieren Sie das zweistellige Prädikat `gewicht`. Es gilt `gewicht(Socke,Gewicht)` genau dann, wenn `Gewicht` das Gewicht aller Geschenke in `Socke` ist. Für die Socke `T` gilt `gewicht(T,250)`.

b) Implementieren Sie das dreistellige Prädikat `einfuegen`. Es gilt

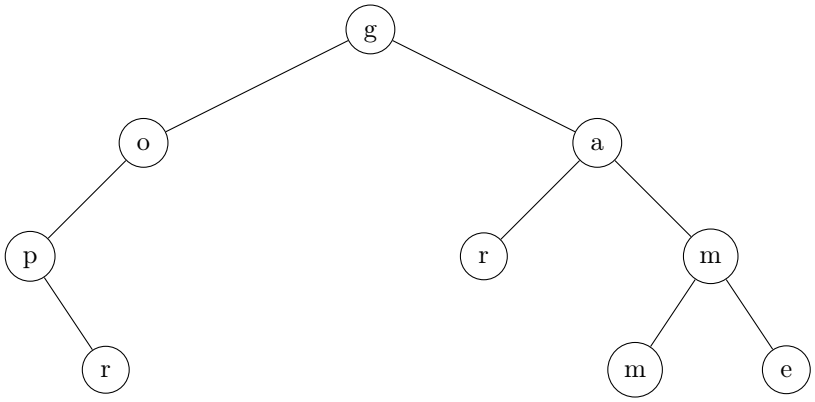
```
einfuegen(SockeVorher,Geschenk,SockeNachher)
```

genau dann, wenn `SockeNachher` aus `SockeVorher` dadurch entsteht, wenn man an der richtigen Stelle `Geschenk` einfügt. Hierbei beachten wir die Gesetze der Schwerkraft, d.h. ein Geschenk fällt an die unterste freie Position in der Socke (ein Geschenk darf also nie über Luft hängen). Für die Socken `S` und `T` gilt `einfuegen(S,geschenk(50, "Schokolade"),T)`.

Aufgabe 6 (Binärbäume): (2 + 2 + 2 + 2 = 8 Punkte)

In dieser Aufgabe sollen Sie Binärbäume in Prolog darstellen und einige Algorithmen darauf implementieren. Verwenden Sie das dreistellige Prädikat `baum(Wert,Links,Rechts)`, um Knoten des Baumes darzustellen. `Wert` bezeichnet dabei den Wert des Knotens, `Links` ist der linke und `Rechts` ist der rechte Teilbaum. Verwenden Sie das nullstellige Prädikat `nil` für die Blätter des Baumes.

a) Geben Sie in Prolog eine Darstellung für den folgenden Baum an.



b) Schreiben Sie die Funktion `baumEnthaelt(X, Baum)`, die genau dann `true` liefert, wenn der Baum `Baum` einen Knoten mit dem Wert `X` enthält. Wenn `B` der Baum aus Aufgabe a) ist, so liefern zum Beispiel `baumEnthaelt(p,B) true` und `baumEnthaelt(q,B) false`.

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

d) 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 Baum aus Teilaufgabe a) ergeben sich folgende Ausgaben:

- Preorder: goprarmme
- Postorder: rpormemag
- Inorder: programme

Sie sollen nun eine Funktion `inorder(B,Res)` schreiben, die alle Knoten eines Baumes in **In**order in die Liste `Res` einträgt. Wenn `B` der Baum aus Teilaufgabe a) ist, so würde `inorder(B,Res)` das Ergebnis `Res = [p,r,o,g,r,a,m,m,e]` liefern.

Hinweise:

- Sie dürfen die Funktion `zusammen` aus Teilaufgabe c) verwenden.

Tutoraufgabe 7 (Arithmetik in Prolog):

Formulieren Sie ein Prolog-Programm mit einem Prädikat `squares(N, R)`, das wahr ist, wenn $N \geq 1$ 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 und benutzen Sie das vordefinierte Prädikat `is/2`.

Aufgabe 8 (Arithmetik in Prolog): (3 Punkte)

Formulieren Sie ein Prolog-Programm mit einem Prädikat `fibs(N, R)`, das wahr ist, wenn $N \geq 2$ gilt und `R` die absteigende Liste der ersten `N` Fibonacci-Zahlen ist. Beispielsweise soll `fibs(7, [13, 8, 5, 3, 2, 1, 1])` wahr sein. Verwenden Sie als Basisfall, dass `fibs(2, [1,1])` gilt und benutzen Sie das vordefinierte Prädikat `is/2`.