Prof. Dr. J. Giesl
M. Brockschmidt, F. Emmes, C. Otto, T. Ströder

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

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

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

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

(v) **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:

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

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

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

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

Lösung:

(i)
$\sigma_1 = \{X = g(Y, Y)\}$
$\sigma_2 = \{Y = g(Z, Z)\}$
$\sigma_3 = \{Z = a\}$
$\sigma = \{X = g(g(a, a), g(a, a)), Y = g(a, a), Z = a\}$

(ii)
$\sigma_1 = \{Y = f(a, X)\}$
$\sigma_2 = \{Z = f(a, X)\}$
$\sigma_3 = \{X = h(a)\}$
$\sigma = \{Y = f(a, h(a)), Z = f(a, h(a)), X = h(a)\}$

(iii)
$\sigma_1 = \{Z = h(X)\}$
$\sigma_2 = \{Y = h(X)\}$

$\sigma_3 = \{X = a\}$
$\sigma_4 = \{\}$
$\sigma = \{Z = h(a), Y = h(a), X = a\}$

(iv) $\sigma_1 = \{Y = g(X)\}$
$\sigma_2 = \{Z = g(X)\}$
σ_3 existiert nicht, da X in $g(X)$ auftritt. Folglich liegt ein *Occur Failure* vor.

(v) $\sigma_1 = \{X = g(Z)\}$
σ_2 existiert nicht, da $g(Z)$ und $h(Y)$ nicht unifizierbar sind. Folglich liegt ein *Clash Failure* vor.

Aufgabe 2 (Unifikation): (6 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(X,Y,X)** und **f(g(Z),h(Z),Z)**

(ii) **f(g(X,Y),g(Y,Z),g(Z,X))** und **f(g(A,a),g(B,B),g(c,C))**

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

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

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

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

Lösung:

(i) $\sigma_1 = \{X = g(Z)\}$
$\sigma_2 = \{Y = h(Z)\}$
$\sigma_3 = \text{mgu}(g(Z), h(Z))$ existiert nicht, da **g(Z)** und **h(Z)** nicht mit dem gleichen Symbol beginnen. Folglich liegt ein *clash failure* vor.

(ii) $\sigma_1 = \{X = A, Y = a\}$
$\sigma_2 = \text{mgu}(g(a, Z), g(B, B)) = \{B = a, Z = a\}$
$\sigma_2 = \text{mgu}(g(a, a), g(c, C))$ existiert nicht, da **a** und **c** nicht unifizieren: *clash failure*

(iii) $\sigma_1 = \{X = a\}$
$\sigma_2 = \text{mgu}(Y, s(a)) = \{Y = s(a)\}$
$\sigma_3 = \text{mgu}(\mathbf{s(a)}, \mathbf{s(Z)}) == \{\mathbf{Z = a}\}$
$\sigma_4 = \text{mgu}(\mathbf{a}, \mathbf{a}) = \emptyset$
$\sigma = \{\mathbf{X = a}, \mathbf{Y = s(a)}, \mathbf{Z = a}\}$

(iv) $\sigma_1 = \{X = Y\}$

$\sigma_2 = \{Z = Y\}$
$\sigma_3 = \text{mgu}(Y, s(Y))$ existiert nicht. *occur failure.*

(v) $\sigma_1 = \{X = Z\}$
$\sigma_2 = \{Z = s(Y)\}$
$\sigma_3 = \text{mgu}(s(Y), s(b)) = \{Y = b\}$
$\sigma_4 = \text{mgu}(b, a)$ existiert nicht. *clash failure.*

(vi) $\sigma_1 = \{X = s(Y)\}$
$\sigma_2 = \text{mgu}(s(Y), s(s(Z))) = \{Y = s(Z)\}$
$\sigma_3 = \{Z = a\}$
$\sigma = \{X = s(s(a)), Y = s(a), Z = a\}$

Tutoraufgabe 3 (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(c,X).
t(X,b) :- t(c,X).
t(c,b).
```

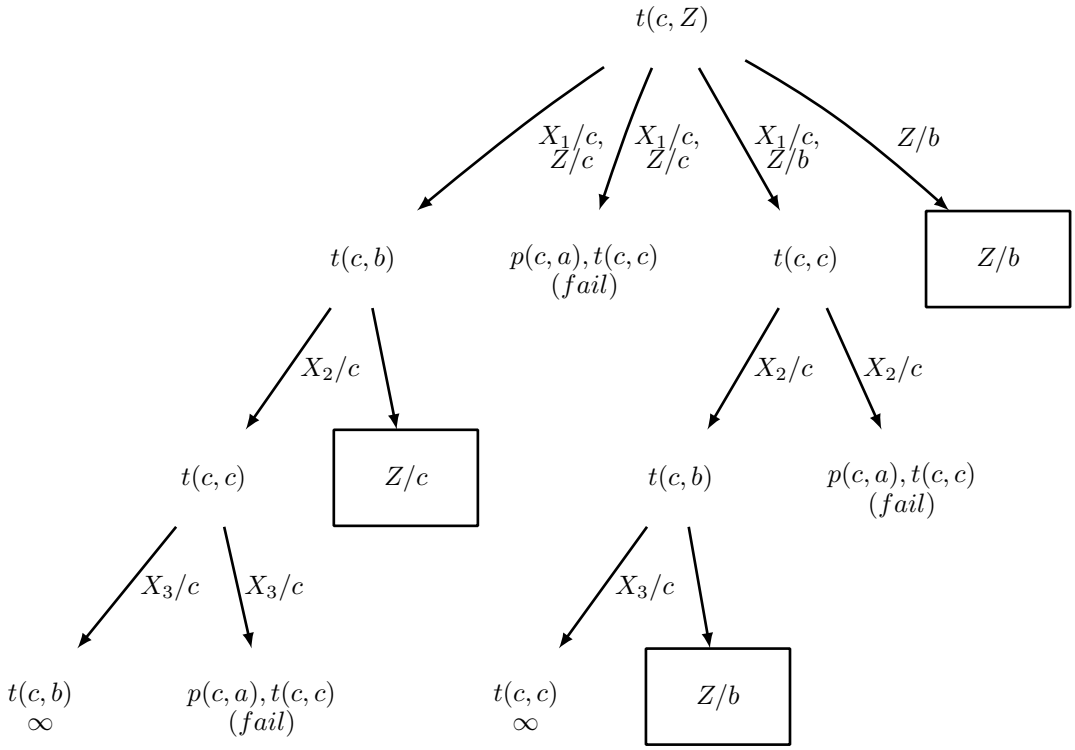
a) Geben Sie den zugehörigen Beweisbaum (SLD-Baum) bis einschließlich Höhe 3. Die Höhe eines Baums ist der längste Pfad von der Wurzel bis zu einem Blatt (ein binärer Baum, welcher nur aus einem Blatt besteht, hat also die Höhe 0). Markieren Sie unendliche Pfade mit ∞ und Fehlschläge mit ($fail$). Geben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

Lösung:

a)

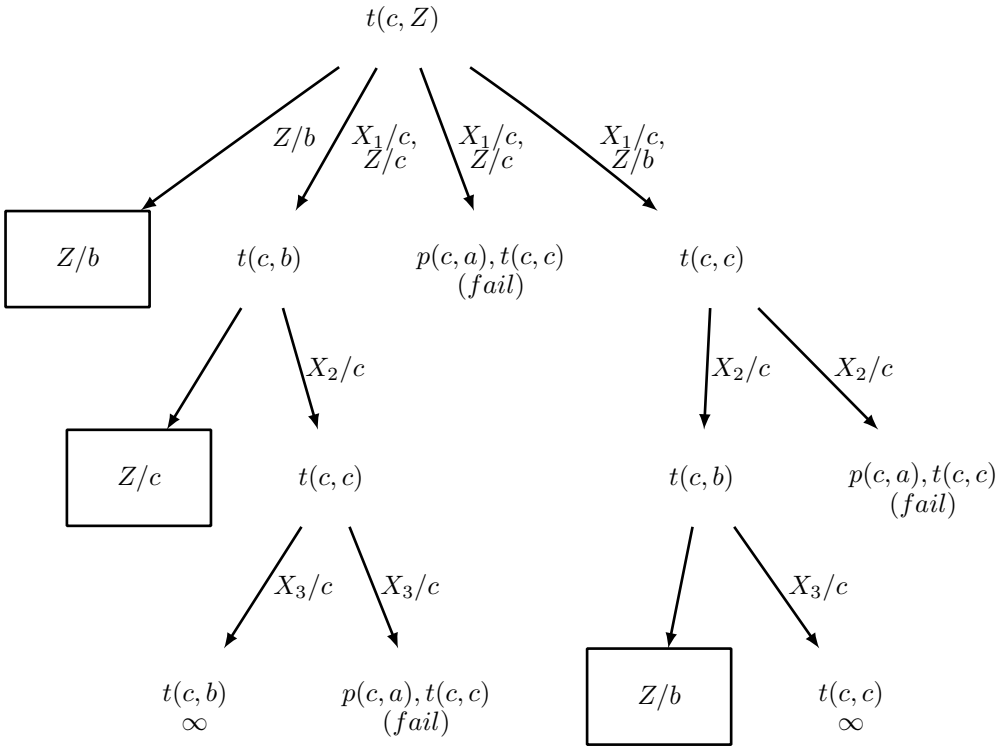


Es gibt (in diesem Teilbaum) drei Antwortsubstitutionen: `Z/c` und zwei mal `Z/b`.Allerdings werden diese nicht erreicht, da der am weitesten links stehende Ast unendlich ist.

b) Das Programm sollte wie folgt umstrukturiert werden:

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

Es ergibt sich dann der folgende, unendliche Beweisbaum für die Anfrage `?- t(c,Z).`:



Der Vorteil gegenüber dem ursprünglichen Programm ist hier, dass der am weitesten links stehende Ast zu einer Lösung führt.

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

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).
```

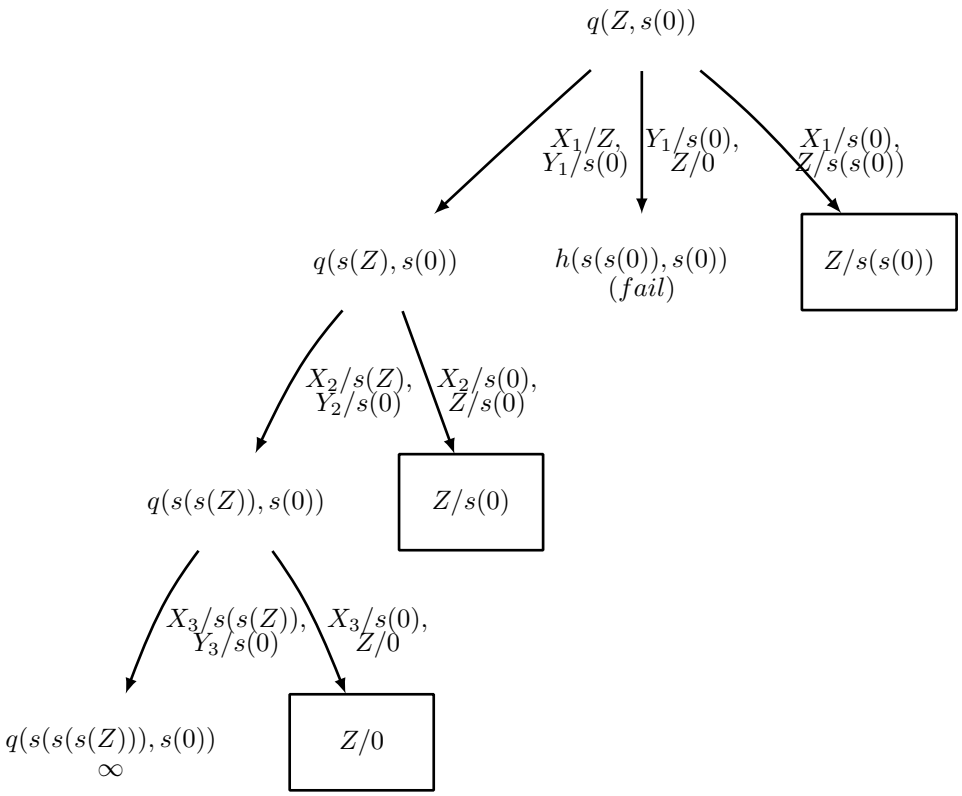
a) Geben Sie den zugehörigen Beweisbaum (SLD-Baum) bis einschließlich Höhe 3. Die Höhe eines Baums ist der längste Pfad von der Wurzel bis zu einem Blatt (ein binärer Baum, welcher nur aus einem Blatt besteht, hat also die Höhe 0). Markieren Sie unendliche Pfade mit ∞ und Fehlschläge mit $(fail)$. Geben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

Lösung:

a)

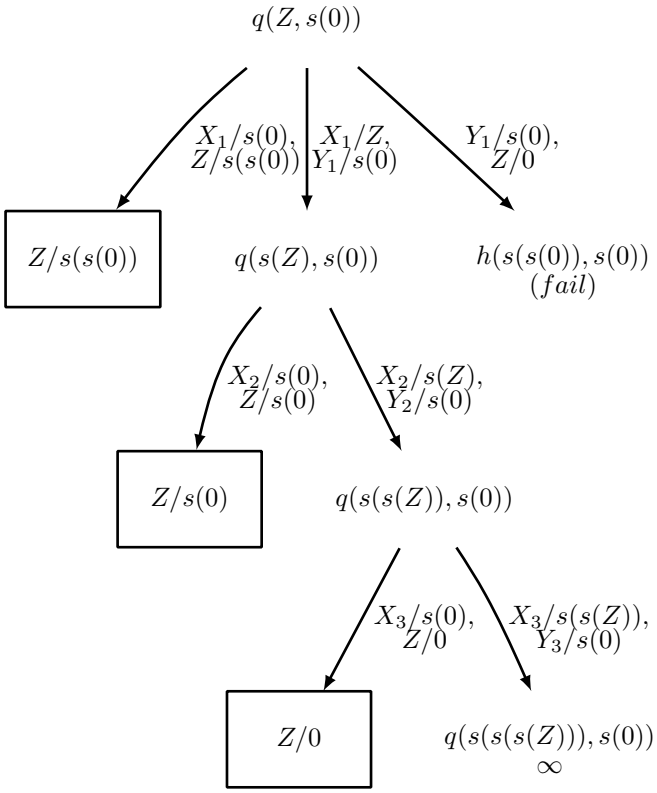


Es gibt drei Antwortsubstitutionen: $Z/0$, $Z/s(0)$ und $Z/s(s(0))$.

b) Das Program kann wie folgt umstrukturiert werden:

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

Es ergibt sich dann der folgende, unendliche Beweisbaum für die Anfrage `?- q(Z,s(0)).`:



Der Vorteil gegenüber dem ursprünglichen Programm ist hier, dass Prolog die Antworten findet, bevor es in den nichtterminierenden Ast der Berechnung läuft.

Tutoraufgabe 5 (Arithmetik in 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 und benutzen Sie das vordefinierte Prädikat `is/2`.

Lösung:

```
squares(1, [1]).
squares(N, [NN,T2|R]) :- N > 1,
                         T is N - 1,
                         squares(T, [T2|R]),
                         NN is T2 + 2*T + 1.
```

Aufgabe 6 (Arithmetik in Prolog): (3 Punkte)

Formulieren Sie ein Prolog-Programm mit einem Prädikat `fibs(N, R)`, das wahr ist, wenn `N` ≥ 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`.

Lösung:

```
fibs(2, [1,1]).
fibs(N, [F,T1,T2|R]) :- N > 2,
                        T is N - 1,
                        fibs(T, [T1,T2|R]),
                        F is T1 + T2.
```