RWTH Aachen **WS 2013/14**
Lehrgebiet Theoretische Informatik Übungsblatt 12
Frohn–Reidl–Rossmanith–Sánchez–Ströder 27.01.2014

Übung zur Vorlesung Programmierung

Aufgabe T26

Gegeben sei die folgende Liste an Relationen, die alle Partnerstädte in Europa darstellt.

```
twin(aachen,halifax).
twin(aachen,naumburg).
twin(aachen,reims).
twin(aachen,toledo).
twin(aalborg,galway).
```

[...]

```
twin(zvolen,niederanven).
twin(zvolen,sherborne).
twin(zvolen,susice).
twin(zweibruecken,boulognesurmer).

adj(X,Y) :- twin(X,Y).
adj(X,Y) :- twin(Y,X).
```

Schreiben Sie ein Prädikat `path(N,X,Y)` in Prolog, das `true` ergibt, wenn es einen Pfad der Länge `N` über benachbarte Städte von `X` nach `Y` gibt. Dafür dürfen Sie das Prädikat `is` verwenden.

Lösungsvorschlag

```
path(0,X,X).
path(N,X,Y) :- N>0, N1 is N-1, adj(X,Z), path(N1, Z, Y).
```

Aufgabe T27

In dieser Aufgabe sollen Unifikatoren bestimmt werden. Nutzen Sie den im Folgenden beschriebenen Algorithmus zur Berechnung eines Unifikators, um die im Anschluss an den Algorithmus gegebenen 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.

Algorithmus MGU zur Berechnung des Unifikators

Eingabe zwei Terme s und t

Ausgabe Unifikator oder $\perp$

1. Falls s und t gleiche Variablen sind, dann $\sigma = \emptyset$.
2. Falls s eine Variable ist und t nicht s enthält, dann $\sigma = \{s = t\}$.
3. Falls t eine Variable ist und s nicht t enthält, dann $\sigma = \{t = s\}$.
4. Falls $s = f(s_1, \ldots, s_n)$ und $t = f(t_1, \ldots, t_n)$, dann:
   4.1. Sei $\sigma_1 = MGU(s_1, t_1)$.
   4.2. Sei $\sigma_i = MGU(\sigma_{i-1} \circ \ldots \circ \sigma_1(s_i), \sigma_{i-1} \circ \ldots \circ \sigma_1(t_i))$ für alle $2 \leq i \leq n$.
   4.3. Falls alle σ_i existieren, dann $\sigma = \sigma_n \circ \ldots \circ \sigma_1$.
5. Sonst $\perp$.

Termpaare

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

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

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

(iv) **h(g(Z,Z), Z)** und **h(X. g(X))**

Lösungsvorschlag

(i)
$$\begin{aligned} \sigma_1 &= \{X = g(a)\} \\ \sigma_2 &= \{A = Z\} \\ \sigma_3 &= \{Z = a\} \\ \sigma_4 &= \{Y = a\} \\ \sigma &= \{A = a, X = g(a), Y = a, Z = a\} \end{aligned}$$

(ii)
$$\begin{aligned} \sigma_1 &= \{X = a\} \\ \sigma_2 &= \{Z = g(a)\} \\ \sigma_3 &= \{Y = b\} \end{aligned}$$
σ_4 existiert nicht, da b und $g(a)$ nicht mit dem gleichen Symbol beginnen. Folglich liegt ein *clash failure* vor.

(iii)
$$\begin{aligned} \sigma_1 &= \{X = g(Y, Z)\} \\ \sigma_2 &= \{Y = Z\} \\ \sigma_3 &= \{Z = a\} \\ \sigma &= \{X = g(a, a), Y = a, Z = a\} \end{aligned}$$

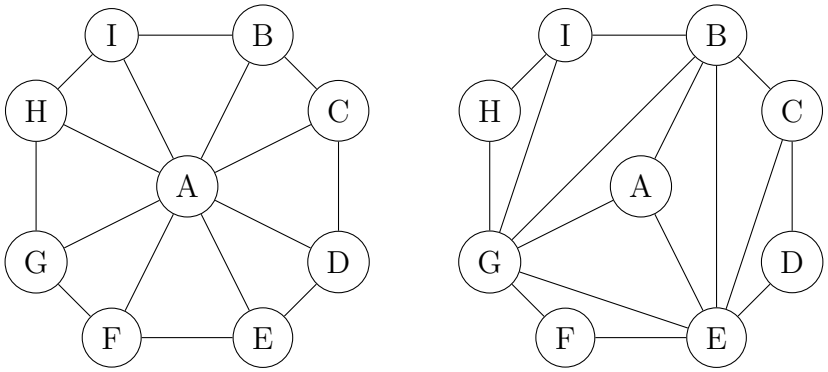
(iv)
$$\sigma_1 = \{X = g(Z, Z)\}$$
σ_2 exisitiert nicht, da Z in $g(g(Z, Z))$ auftritt. Folglich liegt ein *occur failure* vor.

Aufgabe T28

In dieser Aufgabe betrachten wir das Problem der 3-Färbbarkeit von Graphen. Ein Graph ist 3-färbbar, wenn jedem Knoten des Graphen genau eine der drei Farben Rot, Grün oder Blau zugewiesen werden kann, sodass keine zwei benachbarten Knoten die gleiche Farbe haben.

Implementieren Sie dazu zunächst ein Prädikat `verschieden/2` in Prolog, wobei `verschieden(X,Y)` genau dann wahr sein soll, wenn `X` und `Y` zwei verschiedene Konstanten aus der Menge {`rot`, `gruen`, `blau`} sind.

Betrachten Sie außerdem die folgenden beiden Graphen:



Geben Sie für jeden Graphen jeweils eine Anfrage über die Variablen `A`, `B`, `C`, `D`, `E`, `F`, `G`, `H` und `I` an, mit der Prolog alle 3-Färbungen für die Knoten `A`–`I` ausgibt, die der jeweilige Graph besitzt. Nutzen Sie dazu das von Ihnen implementierte Prädikat `verschieden/2`.

Lösungsvorschlag

```
verschieden(rot,blau).
verschieden(rot,gruen).
verschieden(blau,rot).
verschieden(blau,gruen).
verschieden(gruen,rot).
verschieden(gruen,blau).

?- verschieden(A,B),
   verschieden(A,C),
   verschieden(A,D),
   verschieden(A,E),
   verschieden(A,F),
   verschieden(A,G),
   verschieden(A,H),
   verschieden(A,I),
   verschieden(B,C),
   verschieden(B,I),
   verschieden(C,D),
   verschieden(D,E),
   verschieden(E,F),
   verschieden(F,G),
   verschieden(G,H),
```

```
   verschieden(H,I).

?- verschieden(A,B),
   verschieden(A,E),
   verschieden(A,G),
   verschieden(B,C),
   verschieden(B,E),
   verschieden(B,G),
   verschieden(B,I),
   verschieden(C,D),
   verschieden(C,E),
   verschieden(D,E),
   verschieden(E,F),
   verschieden(E,G),
   verschieden(F,G),
   verschieden(G,H),
   verschieden(G,I),
   verschieden(H,I).
```

Aufgabe H32 (5 Punkte)

Gegeben sei die gleiche Liste von Partnerstädten wie in Aufgabe T26. Schreiben Sie Prädikate `triangle`, `fourclique`, `fiveclique`, `sevenclique`. Diese sollen jeweils *true* sein, wenn es eine Clique der Größe 3, 4, 5, 6 oder 7 gibt. Eine Clique der Größe n ist hierbei eine Menge n an Städten, die alle paarweise miteinander verpartnert sind.

Lösungsvorschlag

```
triangle(X,Y,Z) :- adj(X,Y), adj(X,Z), adj(Y,Z).
fourclique(W,X,Y,Z) :- adj(X,W), adj(Y,W), adj(Z,W),
                       triangle(X,Y,Z).
fiveclique(V,W,X,Y,Z) :- adj(X,V), adj(Y,V), adj(Z,V),
                         adj(W,V), fourclique(W,X,Y,Z).
sixclique(U,V,W,X,Y,Z) :- adj(X,U), adj(Y,U), adj(Z,U),
                          adj(W,U), adj(V,U),
                          fiveclique(V,W,X,Y,Z).
sevenclique(T,U,V,W,X,Y,Z) :- adj(X,T), adj(Y,T), adj(Z,T),
                              adj(W,T), adj(V,T), adj(U,T),
                              sixclique(U,V,W,X,Y,Z).
```

Aufgabe H33 (5 Punkte)

In dieser Aufgabe sollen Unifikatoren bestimmt werden. Nutzen Sie den in Aufgabe T27 beschriebenen Algorithmus zur Berechnung eines Unifikators, 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.

(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)**

Beispiele:
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.

$\sigma = \perp$

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

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

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

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

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

Lösungsvorschlag

(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 H34 (5 Punkte)[1]

Sudoku Rätsel sind heutzutage allgemein bekannt. Bei einem Sudoku Rätsel betrachtet man ein $n \times n$ Quadrat von n verschiedenen Ziffern, wobei n selbst auch eine Quadratzahl ist. Dabei muss jede Zeile und jede Spalte jede der n Ziffern genau einmal enthalten. Außerdem muss jedes der n Unterquadrate jede der n Ziffern genau einmal enthalten. Weitere Informationen zu Sudoku erhalten Sie z. B. hier: `http://de.wikipedia.org/wiki/Sudoku`

Um die Aufgabe übersichtlich zu halten, betrachten wir hier lediglich 4×4 Sudokus (es gilt also $n = 4$). Die Form eines solchen Sudokus ist nachfolgend dargestellt:

X11	X12	X13	X14
X21	X22	X23	X24
X31	X32	X33	X34
X41	X42	X43	X44

Die vier Unterquadrate sind hier (`X11`, `X12`, `X21`, `X22`), (`X13`, `X14`, `X23`, `X24`), (`X31`, `X32`, `X41`, `X42`) und (`X33`, `X34`, `X43`, `X44`).

Implementieren Sie zunächst ein Prädikat `permutation/4` in Prolog, wobei `permutation(A,B,C,D)` genau dann wahr sein soll, wenn (`A`, `B`, `C`, `D`) eine Permutation der Folge $(1, 2, 3, 4)$ ist (also diese vier Ziffern in beliebiger Reihenfolge, aber jede der vier Ziffern muss genau einmal vorkommen).

Geben Sie außerdem eine Anfrage über die Variablen `X11`, `X12`, `X13`, `X14`, `X21`, `X22`, `X23`, `X24`, `X31`, `X32`, `X33`, `X34`, `X41`, `X42`, `X43` und `X44` an, mit der Prolog alle vollständig ausgefüllten gültigen Sudokus für $n = 4$ der oben dargestellten Form ausgibt.

Hinweis: Es gibt 288 gültige Sudokus für $n = 4$. Sie sollten also nur die Implementierung Ihres Prädikats und die Anfrage als Lösung abgeben - nicht die erhaltenen Antworten.

Lösungsvorschlag

```
permutation(1,2,3,4).
permutation(1,2,4,3).
permutation(1,3,2,4).
permutation(1,3,4,2).
permutation(1,4,2,3).
permutation(1,4,3,2).
permutation(2,1,3,4).
permutation(2,1,4,3).
permutation(2,3,1,4).
permutation(2,3,4,1).
permutation(2,4,1,3).
permutation(2,4,3,1).
permutation(3,1,2,4).
```

```
permutation(3,1,4,2).
permutation(3,2,1,4).
permutation(3,2,4,1).
permutation(3,4,1,2).
permutation(3,4,2,1).
permutation(4,1,2,3).
permutation(4,1,3,2).
permutation(4,2,1,3).
permutation(4,2,3,1).
permutation(4,3,1,2).
permutation(4,3,2,1).

?- permutation(X11,X12,X13,X14),
   permutation(X21,X22,X23,X24),
   permutation(X31,X32,X33,X34),
   permutation(X41,X42,X43,X44),
   permutation(X11,X21,X31,X41),
   permutation(X12,X22,X32,X42),
   permutation(X13,X23,X33,X43),
   permutation(X14,X24,X34,X44),
   permutation(X11,X12,X21,X22),
   permutation(X13,X14,X23,X24),
   permutation(X31,X32,X41,X42),
   permutation(X33,X34,X43,X44).
```

Bewertungsvorschlag: 2 Punkte für `permutation/4`, 3 Punkte für Anfrage.

Abgabe zum 04.02.2014

[1] Bitte Quelltext für die Abgabe ausdrucken und zusätzlich per E-mail an den jeweiligen Tutor senden.