Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Informatik

Programming Concepts and Paradigms (PCP)

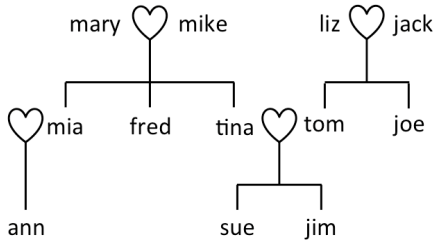
PCP-Übung zu Prolog 1 + 2

Hauptthemen: Prolog Syntax, Prädikate, Matching & Backtracking
Zeitfenster: ca. 2-3 Lektionen

Ruedi Arnold

Diese Übung repetiert und vertieft den in Prolog 1 und 2 vermittelten Stoff. Gehen Sie zur Vorbereitung nochmals durch die Unterrichtsfolien durch und schauen Sie insbesondere, dass Sie alle auf den Folien angegebenen Code-Beispiele verstanden haben.

1. Verwandtschaftsbeziehungen



Bei dieser Aufgabe geht es um Verwandtschaftsbeziehungen, gegeben ist der unvollständige Stammbaum einer Familie im Bild rechts. Mary und Mike haben also drei Kinder Mia, Fred und Tina, usw. Diese gesamten Informationen sind in folgenden Fakten mithilfe von den drei selberdefinierten Prolog-Prädikaten female/1, male/1 und parent/2 wiedergegeben:

```
female(mary). female(liz). female(mia). female(tina). female(ann). female(sue).% all females
male(mike). male(jack). male(fred). male(tom). male(joe). male(jim).           % all males
parent(mary, mia). parent(mary, fred). parent(mary, tina).                     % all childern of mary
parent(mike, mia). parent(mike, fred). parent(mike, tina).                     % all children of mike
parent(liz, tom). parent(liz, joe).                                            % allchildern of liz
parent(jack, tom). parent(jack, joe).                                          % all childern of jack
parent(mia, ann).                                                              % all childern of mia
parent(tina, sue). parent(tina, jim).                                          % all childern of tina
parent(tom, sue). parent(tom, jim).                                            % all childern of tom
```

Verwenden Sie in der folge diese Fakten, um darauf basierend folgende Regeln zu erstellen und überprüfen.

a) Implementieren Sie zwei Prädikate mother/2 und father/2. Die Anfrage `mother(X, tina)` soll dann also beispielsweise `X = mary` zurückliefern und die Anfrage `father(X, sue)` entsprechend `X = tom`. Verwenden Sie dieses beiden Prädikate in der Folge beispielsweise, um zu überprüfen, wer die Mutter und der Vater von jim ist. Und mit welcher Abfrage können Sie herausfinden, wie alle Kinder von Mary heissen?

b) Implementieren Sie ein weiteres Prädikat sibling/2, welches überprüft, ob zwei Personen Geschwister sind. Auf die Anfrage `sibling(mia, fred)` wird also beispielsweise die Antwort `true` erwartet. Hinweis 1: Da wir bisher keine Negation kennen, ist es ok, wenn eine Person ein Geschwister von sich selbst ist, wenn also z.B. das System auf `sibling(sue, sue)` mit `true` antwortet. Negation, also wie wir z.B. sagen können „nicht(X=Y)“, schauen wir später an.

c) Implementieren Sie ein Prädikat grandmother/2. Die Anfrage `grandmother(X, ann)` soll dann also beispielsweise `X = mary` zurückliefern und die Anfrage `grandmother(liz, X)` entsprechend `X = sue; X = jim`. Lassen Sie sich mit diesem Prädikat alle Grossmütter von jim ausgeben. Wie lautet die entsprechende Anfrage?

d) Implementieren Sie zum Schluss ein Prädikat offspring/2, welche erfüllt ist, wenn Person 1 (erstes Argument) ein Nachkomme von Person 2 (zweites Argument) ist. Auf die Anfrage `offspring(ann, mary)` soll das System also mit `true` antworten. Oder auf die Anfrage `offspring(sue, X)` (Bedeutung: wer sind alle Vorfahren von Sue?) soll das System antworten mit `X = tina; X = tom; X = mary; X = mike; X = liz; X = jack`. Hinweis: Dieses Prädikat ist rekursiv und definiert eine transitive Hülle, analog zum Beispiel is_bigger/2 von den Unterrichtsfolien.

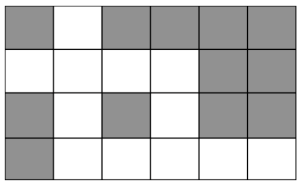
2. Matching & Suchbaum

Die Prädikate male/1, female/1, parent/2, mother/2 und grandmother/2 sind wie bei Aufgabe 1 definiert.

a) Zeichnen Sie den kompletten Suchbaum auf, welcher auf die Anfrage `grandmother(liz, X)` aufgebaut und durchsucht wird.

b) Zeichnen Sie den kompletten Suchbaum auf, welcher auf die Anfrage `offspring(ann, mary)` aufgebaut und durchsucht wird.

3. Kreuzworträtsel lösen



In Unterricht wurde gezeigt, wie mit Hilfe von Prolog einfach Kreuzworträtsel gelöst werden können. Lösen Sie analog zum im Unterricht gezeigten Beispiel das rechts angegebene Kreuzworträtsel mit Hilfe von einem Prolog-Programm. Die zur Verfügung stehenden Wörter sind: neu, top, tot, brot, grau, halt, alle, jetzt, sagen, unten, zecke.

4. Karte färben

Analog zum Beispiel im Unterricht sollen die sechs Zentralschweizer Kantone LU, NW, OW, SZ, UR und ZG (siehe Bild rechts) mit den drei Farben gelb, rot und grün eingefärbt werden. Dabei dürfen angrenzende Kantone nicht dieselbe Farbe haben. Weiter ist vorgegeben, dass der Kanton Uri gelb sein soll und der Kanton Schwyz rot. Lösen Sie diese Aufgabe mit Hilfe von einem Prolog-Programm. Wie viele Lösung gibt es zu dieser Problemstellung?

5. Eigene Familien-Operatoren definieren

Ausgangslage sind die in Aufgabe 1 angegeben und von ihnen erstellten Fakten und Regeln zu Verwandtschaftsbeziehungen.

a) Implementieren Sie basierend auf dem Prädikat mother/2 einen xfx-Operator. Die Anfrage `liz mother X` soll dann also beispielsweise `X = tom; X = joe` zurückliefern.

b) Implementieren Sie basierend auf dem Prädikat offspring/2 ebenfalls einen xfx-Operator. Die Anfrage `ann offspring mike` soll dann also beispielsweise `true` zurückliefern.

6. Aufgabe: Operatoren und arithmetische Ausdrücke

a) Wie wird in SWI-Prolog der Term `X is 16 / 4 / 2` ausgewertet? Erklären sie, weshalb das so ist.

b) Was ist die Antwort auf die Anfrage `Y = 3, X = Y - 1`?

c) Was ist die Antwort auf die Anfrage `Y = 3, X is Y - 1`? Wie lässt sich das unterschiedliche Resultat gegenüber der Teilaufgabe b) erklären?

7. Aufgabe: Multiplikation durch rekursive Addition

a) Implementieren Sie ein eigenes Prädikat mult/3, welches erfüllt ist, wenn die ersten beiden Argumente miteinander multipliziert, dem Wert des dritten Arguments entsprechen. Dazu dürfen Sie den Multiplikationsoperator nicht verwenden, sondern Sie sollen multiplizieren durch wiederholtes addieren. Der Aufruf von `mult(3, 4, X)` soll also beispielsweise `X = 12` zurück liefern und `mult(3, 4, 11)` entsprechend `false`. Sie dürfen davon ausgehen, dass die ersten beiden Argumente >= 0 sind. Hinweis: Achtung, mult/3 soll auch funktionieren, falls eines der ersten zwei oder die beide ersten Argumente 0 oder 1 sind, also z.B. für die Fälle `mult(3, 0, X)`, `mult(0, 3,0)` oder `(77, 1, X)`.

b) Wie können Sie verhindern, dass mult/3 den Fehler „Out of local stack“ produziert? Dies passiert beispielsweise, wenn Sie sich vom System beim Aufruf von `mult(3, 4, X)` alle Lösungen für `X` angeben lassen (indem Sie wiederholt ; drücken). Wie entsteht dieser Fehler? Verbessern Sie ihr Prädikat mult/3 so, dass dieser Fehler nicht mehr auftreten kann.