Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Institut für Informatik

1. Übung zur Vorlesung „Deduktive Datenbanken und logische Programmierung“

Wintersemester 2007/2008 Ausgabe: 2007-10-15

Abgabe: 2007-10-22

Für die Übungen benutzen wir SWI-Prolog (`http://www.swi-prolog.org/`).

Aufgabe 1.1: Mathematiker-Stammbaum

Wer nach dem Studium an der Universität bleibt, schreibt in der Regel eine Doktorarbeit. Bei dieser Arbeit wird sie oder er von einem Professor betreut, dem Doktorvater. Manche Doktoranden werden später selbst Professor und betreuen ihrerseits Doktoranden. Durch diese Beziehungen entsteht eine Art Ahnenbaum, der sehr verästelt sein und sich über mehrere Kontinente erstrecken kann. Als Fakten sind atomare Aussagen in folgender Form `doctorate(ID, Name, Degree, University, Year, Country, KindOfThesis, Thesis)` und `advisor(ID, AdvisorID)` gegeben (siehe WWW-Seite der Übung). Die folgenden Aufgaben sind ausschließlich mit Prolog-Anfragen zu lösen, d.h. zu den abzugebenden Lösungen müssen alle Prolog-Anfragen beigelegt werden.

a) Schreiben Sie ein logisches Programm in Prolog, mit dem Sie den Ahnenbaum für einen gegebenen Namen eines Doktoranden rekonstruieren können. Beachten Sie, daß manche Doktoranden zwei oder mehrere Betreuer hatten.

b) Stellen Sie den Ahnenbaum von Harold Stephen Finkelstein graphisch dar.

c) Schreiben Sie ein Programm, das die akademischen Nachkommen von Gustav Peter Lejeune Dirichlet ausgibt, die in den Vereinigten Staaten nach 1959 promoviert wurden.

Hinweis: In Prolog werden in Anführungszeichen eingeschlossene Texte in Listen von ASCII-Codes übersetzt. Mit dem Prädikat `string_to_list` können sie diese Listen zu Texten oder zurück konvertieren:

```
string_to_list(String, List)     .
```

Siehe auch die Beschreibung von SWI-Prolog.

e-mail: {brass,thielema}@informatik.uni-halle.de Web: http://www.informatik.uni-halle.de/~{brass,thielema}

Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Institut für Informatik

2. Übung zur Vorlesung „Deduktive Datenbanken und logische Programmierung“

Wintersemester 2007/2008 Ausgabe: 2007-10-22

Abgabe: 2007-10-29

Aufgabe 2.1: Mathematiker-Stammbaum

a) Gibt es Doktoranden, die vor ihrem Doktorvater promoviert wurden?

b) Berechnen Sie zu jedem mittelbaren Doktorvater von Harold Stephen Finkelstein die Anzahl der dazwischenliegenden akademischen Generationen. Beispiel: Zwischen Finkelstein und Finkelstein liegen null Generationen, zwischen Finkelstein und seinen direkten Doktorvätern liegt eine Generation.
Hinweis: Die Aussage

```
Z is X+Y
```

bedeutet, dass die Summe rechts von `is` berechnet und das Ergebnis mit der linken Seite abgeglichen (sog. „unification“) wird.

c) Schreiben Sie ein Prädikat für Doktoranden mit mehr als einem direkten Doktorvater.

d) Schreiben Sie ein Prädikat für die Anzahl der (direkten) Doktoranden eines Doktorvaters. Benutzen Sie hierzu die Aussageformen

```
length(List, Len)
findall(X, pred(X), ListOfMatchingXs)
```

e) Finden Sie den Doktorvater mit den meisten (direkten) Doktoranden.

   Tipp: Nutzen Sie ein Hilfsprädikat für die Doktorväter, auf die das nicht zutrifft.

   Falls Ihr Rechner zu lange für diese Aufgabe braucht, überlegen Sie sich, wie Sie die Aufgabe interaktiv im Prolog-Interpreter lösen können.

f) Denken Sie sich selbst eine Anfrage aus, die Sie in Prolog programmieren.

Aufgabe 2.2: Typabgleich

Das Typsystem der Programmiersprache Haskell kennt Typvariablen und kann selbstständig deren Belegung ermitteln. Beispielsweise wendet die Funktion `map` eine andere Funktion elementweise auf Listenelemente an und die Funktion `toUpper` wandelt Buchstaben in Großbuchstaben um. Folglich wandelt `map toUpper` eine Liste von Buchstaben in eine Liste von Großbuchstaben um. Aus den Typsignaturen

```
map :: (a -> b) -> ([a] -> [b])
toUpper :: Char -> Char
```

schließt das Typsystem selbständig, dass `map toUpper` den Typ `[Char] -> [Char]` besitzt. Die Typvariablen `a` und `b` werden beide mit dem konkreten Typ `Char` belegt.
Diese Typbestimmung lässt sich direkt in Prolog übersetzen. Die Typsignaturen werden wie folgt umschrieben

```
typeMap(func(func(X,Y), func(list(X),list(Y)))).
typeToUpper(func(char, char)).
```

und die Abfrage, welche konkreten Typen sich ergeben, könnte zum Beispiel so aussehen:

```
?- typeMap(func(F, G)), typeToUpper(F).

F = func(char, char)
G = func(list(char), list(char)) ;
```

a) Definieren Sie Prädikate für die Typen folgender Funktionen:

- `reverse` – kehrt die Reihenfolge der Elemente einer Liste um
- `append` – hängt zwei Listen aneinander (benutzen Sie die Schönfinkel/Curry-Form)
- `concat` – hat als Parameter eine Liste von Listen und hängt diese aneinander
- `sum` – addiert eine Liste ganzer Zahlen

Führen Sie falls nötig neue Atome für Typen oder Typkonstruktoren ein.

b) Bestimmen Sie mit Prolog die Typen folgender Ausdrücke:

- `map sum`
- `map reverse`
- `sum (map toUpper x)`

c) Bestimmen Sie mit Prolog die Belegung der Typvariablen der verwendeten Funktionen in folgenden Ausdrücken:

- `map toUpper (concat x)`
- `append x (reverse x)`
- `sum (reverse x)`

Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Institut für Informatik

3. Übung zur Vorlesung „Deduktive Datenbanken und logische Programmierung“

Wintersemester 2007/2008 — Ausgabe: 2007-10-29

Abgabe: 2007-11-05

Aufgabe 3.1: Logik

Geben Sie für folgende Aussagen logische Formeln an.

a) Die Zahlen x, y und z sind aufeinanderfolgende ganze Zahlen.

b) Die Menge A enthält höchstens ein Element.

c) Die Menge A enthält mindestens ein Element.

d) Die Menge A ist leer.

e) Die Menge A enthält genau ein Element.

f) Die Menge A enthält genau zwei Elemente.

g) x ist das Maximum aller Elemente der Menge A.

h) Die Funktion f ist umkehrbar.

i) Die Funktion f nimmt an der Stelle x ihr Minimum an.

j) p ist eine Primzahl.

Aufgabe 3.2: Musiksammlung

Gegeben sei die aus „Datenbanken I“ bekannte CD-Datenbank, jetzt als Prolog-Fakten:

- `komponist(KNR, NAME, VORNAME, GEBOREN, GESTORBEN).`
- `stueck(SNR, KNR→komponist, TITEL, TONART, OPUS).`
  (`TONART` und `OPUS` können Null sein, dargestellt als `'?'`.)
- `cd(CDNR, NAME, HERSTELLER, ANZ_CDS, GESAMTSPIELZEIT).`
- `aufnahme(CDNR→cd, SNR→stueck, ORCHESTER, LEITUNG).`
  (`ORCHESTER` und `LEITUNG` können Null sein, dargestellt als `'?'`).
- `solist((CDNR, SNR)→aufnahme, NAME, INSTRUMENT).`

Sie finden die Daten unter `http://www.informatik.uni-halle.de/~brass/lp07/cd.pro`
Formulieren Sie die folgenden Anfragen in Prolog, entweder direkt als Prolog-Anfrage, oder als Prädikat. Sie können natürlich nach Bedarf Hilfsprädikate deklarieren.

a) Was sind Geburts- und Todesjahr von Wolfgang Amadeus Mozart?

b) Welche CDs enthalten Stücke von Händel? Der Name wird in der Datenbank `'H"andel'` geschrieben. Geben Sie Nummer und Name der CD aus. Auch CDs, die neben Stücken von Händel auch Stücke anderer Komponisten enthalten, sollen ausgegeben werden.

   Bei dieser Aufgabe werden Sie Duplikate erhalten. Sie können die Duplikate auf folgende Weise entfernen: Definieren Sie sich zuerst die Aussagenform `haendel_cd_dup(CD)`, welche die korrekte Lösung mit Duplikaten berechnet. Dann geben Sie folgende Regel ein:

```
haendel_cd(CD) :- setof(CDdup, haendel_cd_dup(CDdup), CD).
```

   Der Aufruf `setof(A, B, C)` setzt `C` auf die Liste aller `A`s, für die `B` gilt, und zwar ohne Duplikate, d.h. $\texttt{C} = \{\texttt{A} \mid \texttt{B}\}$.

c) Geben Sie alle Stücke in C-dur oder a-moll aus. Drucken Sie jeweils den Nachnamen des Komponisten und den Titel des Stücks.

d) Welche CDs (Nummer und Name) enthalten Stücke von mindestens zwei verschiedenen Komponisten? (Sie können `\=` zum Test auf Ungleichheit verwenden.) Sie können Duplikate wie unter b) entfernen.

e) Geben Sie Vorname und Nachname aller Komponisten aus, die von 1700 bis 1799 geboren sind. Vergleiche werden in Prolog `<`, `=<`, `>=`, `>` geschrieben.

f) Geben Sie Name, Vorname und Alter aller Komponisten aus, d.h. jeweils die Differenz von Todesjahr und Geburtsjahr. (Sie können in Prolog arithmetische Ausdrücke mit dem Prädikat `is` auswerten, z.B. `X is Y-Z`.)

g) Was ist der früheste Komponist in der Datenbank, d.h. der mit minimalem Geburtsjahr? (Definieren Sie sich zuerst ein Hilfsprädikat für die nicht-frühesten Komponisten, und verwenden Sie dann `not` oder `\+`, um zu testen, dass dieses Prädikat für den auszugebenen Komponisten nicht herleitbar ist.) Geben Sie Vorname, Nachname, und Geburtsjahr aus.

Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Institut für Informatik

4. Übung zur Vorlesung „Deduktive Datenbanken und logische Programmierung“

Wintersemester 2007/2008 Ausgabe: 2007-11-05

Abgabe: 2007-11-12

Aufgabe 4.1: Überquerung einer Brücke

Die Gruppe ____________________[1] ist auf dem Weg zu einem Auftritt. Von ihrem Bühnenauftritt trennen sie nur noch 17 Minuten und eine tiefe Schlucht. Es gibt eine wacklige Brücke, über die maximal zwei Leute gleichzeitig und auch nur mit einer Taschenlampe gehen können. Davon haben die Reisenden genau eine. Die Gruppe besteht aus 4 Mitgliedern, wovon einer 1 Minute zum Überqueren der Brücke benötigt, ein anderer 2 Minuten, der dritte 5 Minuten und der letzte 10 Minuten. Wie schafft es die Gruppe, rechtzeitig beim Auftritt zu sein?
Hinweis: Möglicherweise hilft ein neues Prädikat namens `cross`

```
cross(A, (L0,R), (L1,[A|R])) :- select(A, L0, L1).
```

das man wie folgt benutzen kann:

```
?- cross(A, ([1,2,5],[10]), XY).

A = 1
XY = [2, 5], [1, 10] ;

A = 2
XY = [1, 5], [2, 10] ;

A = 5
XY = [1, 2], [5, 10] ;

No
```

Ebenso kann sich die Standardfunktion `max` als recht nützlich erweisen.

Aufgabe 4.2: Wahrheitstabelle

a) Entwickeln Sie in PROLOG ein Prädikat namens `iff`, welches genau dann wahr ist, wenn beide Operanden den gleichen Wahrheitswert besitzen.

b) Entwickeln Sie in PROLOG ein Prädikat, welches testet, ob eine Formel ein Modell hat.

```
?- instantiate((X,(Y;Z)), [X,Y,Z]).

X = true
Y = fail
Z = true ;

X = true
```

[1] bitte gewünschte Gruppe mit 4 Mitgliedern eintragen

```
Y = true
Z = fail ;

X = true
Y = true
Z = true ;
```

Testen Sie es an folgenden Aussagen:

- $\neg p \wedge p$
- $\neg p \vee p$
- $p \wedge (p \vee q)$

c) Entwickeln Sie in PROLOG ein Prädikat, das Formeln darauf testet, ob sie Tautologien darstellen.

```
?- check((not(X);X), [X]).

X = _G183 ;
```

Beweisen Sie mit diesem Prädikat folgende Aussagen:

- $\neg(p \wedge q) \iff \neg p \vee \neg q$
- $p \wedge (p \vee q) \iff p$
- $p \vee (p \wedge q) \iff p$
- $p \wedge (q \vee r) \iff (p \wedge q) \vee (p \wedge r)$

Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Institut für Informatik

5. Übung zur Vorlesung „Deduktive Datenbanken und logische Programmierung“

Wintersemester 2007/2008 Ausgabe: 2007-11-12

Abgabe: 2007-11-19

Aufgabe 5.1: Symbolisches Differenzieren

Definieren Sie ein Prädikat namens `derive`, welches einer Funktion ihre Ableitung zuordnet.
Machen Sie sich selbst Gedanken, wie das am besten zu bewerkstelligen ist. Es sind sehr unterschiedliche Herangehensweisen möglich. Wenn Sie keine Idee haben, folgen Sie folgender Anleitung.
Verarbeiten Sie tatsächlich Ausdrücke von Funktionen, das heißt, dass Funktionsargumente nicht auftauchen. Man schreibt also nicht `x*(log(x)-1)`, sondern `id*(log-1)`. Dabei stehen die Zeichen für die Grundrechenarten für die entsprechenden punktweise definierten Funktionsverknüpfungen. Zum Beispiel steht `(f+g)(x)` für `f(x) + g(x)`. Zahlen stehen für konstante Funktionen, so steht etwa `1` für die Funktion, die überall den Wert `1` annimmt.
Definieren Sie die Ableitungen für die Grundrechenarten (einschließlich Potenzieren :-), für die Identitätsfunktion `id`, für die Exponentialfunktion `exp`, den natürlichen Logarithmus `log`, die Winkelfunktionen `sin`, `cos`, `tan`, und was Ihnen sonst noch so einfällt.
Definieren Sie Ableitungsregeln für die Funktionsverkettung und für Umkehrfunktionen. Es soll `f o g` für die Funktionsverkettung stehen (`(f o g)(x) = f(g(x))`), und `inv(f)` für die Umkehrfunktion von `f`.
Die Ausdrücke müssen nicht vereinfacht werden und es können auch mehrere (richtige) Antworten ausgegeben werden.
Beispiel:

```
?- derive(id*(log-1), Z).

Z = 1* (log-1)+id* (1/id-0) ;

No
```

Hinweis: Das Prädikat `number` stellt fest, ob ein Ausdruck einem Zahlenliteral entspricht.

a) Differenzieren Sie folgende Funktionen

- $x \mapsto x^2$
- $\frac{\sin}{\cos}$
- $\exp \circ \log$
- $x \mapsto \sqrt{1+x^2}$
- $x \mapsto x^x$

b) Integrieren Sie folgende Funktionen

- $\frac{1}{\mathrm{id}}$
- $\exp + \cos$

e-mail: {brass,thielema}@informatik.uni-halle.de Web: http://www.informatik.uni-halle.de/~{brass,thielema}

Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Institut für Informatik

6. Übung zur Vorlesung „Deduktive Datenbanken und logische Programmierung“

Wintersemester 2007/2008 — Ausgabe: 2007-11-19

Abgabe: 2007-11-26

Aufgabe 6.1: Pseudoinverse

In der linearen Algebra wird die Pseudoinverse oder Penrose-Moore-Inverse als Verallgemeinerung der inversen Matrix eingeführt. Im Gegensatz zur inversen Matrix ist sie auch für nicht quadratische und singuläre Matrizen definiert und zwar wie folgt:
Für eine Matrix A wird die Matrix B genau dann Pseudoinverse genannt, wenn sie die Bedingungen

- $A \cdot B = (A \cdot B)^T$
- $B \cdot A = (B \cdot A)^T$
- $A \cdot B \cdot A = A$
- $B \cdot A \cdot B = B$

erfüllt.
Bemerkung: Die ersten beiden Bedingung sind Verallgemeinerungen von $A \cdot A^{-1} = I, A^{-1} \cdot A = I$, wobei I offensichtlich auch symmetrisch ist. Die beiden letzten Bedingung sind ebenfalls solche Verallgemeinerungen. Es wird aber nicht wie bei inversen Matrizen $A \cdot B \cdot x = x$ für alle Vektoren x gefordert, sondern nur für Spaltenvektoren von A.
Der Vektor $B \cdot y$ ist derjenige Vektor x mit kleinster euklidischer Norm, der den Abstand $\|A \cdot x - y\|_2$ minimiert.

$$\begin{aligned} M &= \{x : \forall z \ \|A \cdot x - y\|_2 \leq \|A \cdot z - y\|_2\} \\ B \cdot y &= \operatorname*{argmin}_{x \in M} \|x\|_2 \end{aligned}$$

a) Schreiben Sie ein PROLOG-Prädikat, welches prüft, ob die Pseudoinverse eindeutig definiert ist.

b) Erweitern Sie das Prädikat so, dass es auch den Beweis ausgibt.

Hinweise: Gehen Sie von zwei Matrizen B und C aus, die beide die Bedingungen für Pseudoinverse erfüllen und zeigen Sie, dass $B = C$ gilt, indem Sie die Bedingungen für die Pseudoinverse als Umformungsschritte benutzen. Da die Matrixmultiplikation assoziativ ist, lassen sich alle Terme als Liste von Matrizen verwalten. Implementieren Sie ein Prädikat `replace` für eine „Suchen&Ersetzen“-Operation. Diese lässt sich elegant und ohne Rekursion mit dem Standardprädikat `append` implementieren!
Da die Anzahl der möglichen Terme exponentiell mit der Suchtiefe wächst, sollten Sie sich überlegen, wie sie die Suchtiefe halbieren können. (Sprich: Für einen Beweis mit n Umformungsschritten muss man nur bis zur Rekursionstiefe $\frac{n}{2}$ suchen). Außerdem sollten Sie zur Geschwindigkeitssteigerung überflüssige Terme entfernen. Sollten Sie keinen Beweis finden, haben Sie wahrscheinlich gewisse Einsatzmöglichkeiten der 4 Bedingungen vernachlässigt.
Bemerkung: Für eine reguläre Matrix ist die Pseudoinverse die gewöhnliche Matrixinverse, denn die gewöhnliche Matrixinverse erfüllt die Bedingungen der Pseudoinversen und die Pseudoinverse ist eindeutig definiert.

e-mail: `{brass,thielema}@informatik.uni-halle.de` Web: `http://www.informatik.uni-halle.de/~{brass,thielema}`