Prof. Dr. Stefan Brass
Institut für Informatik
MLU Halle-Wittenberg

23.05.2024

Logische Programmierung & deduktive Datenbanken — Übungsblatt 6 (SLD Resolution, Zahlenrätsel) —

Ihre Lösungen zu den Hausaufgaben 1 und 2 geben Sie bitte über die Übungs-Plattform in StudIP ab. Einsendeschluss ist Mittwoch, der 29. Mai, 16^{00} (die Aufgaben werden in der Übung am Donnerstag besprochen).

Hausaufgaben

Aufgabe 1 (Hausaufgabe, 5 Punkte)

Ein bekannter Typ von Rätsel ist:

```
 SEND
+MORE
-----
MONEY
```

Man muss Ziffern `0...9` für die Buchstaben (Variablen) finden, so dass

- Die Additionsaufgabe korrekt ist,
- verschiedene Buchstaben mit unterschiedlichen Ziffern belegt werden,
- die ersten Buchstaben in jeder Zeile nicht mit `0` belegt werden.

Eine simple Formulierung dieses Rätsels in Prolog ist:

```
generate([S,E,N,D,M,O,R,Y]),
S \= 0,
M \= 0,
to_int([S,E,N,D], I1),
to_int([M,O,R,E], I2),
to_int([M,O,N,E,Y], I3),
I3 =:= I1 + I2.
```

Lösen Sie folgende Teilaufgaben:

a) Definieren Sie zunächst das Hilfsprädikat `generate(L)` zum Einsatz im obigen Programmstück. Dieses Prädikat soll alle Elemente der Liste `L` mit unterschiedlichen Ziffern belegen (auf alle möglichen Arten). Die Ziffern sollen dabei als ganze Zahlen mit Wert zwischen 0 und 9 repräsentiert werden (nicht als Atome/Zeichen). Die Liste `L` besteht aus Variablen oder Ziffern.

b) Als nächstes definieren Sie das Hilfsprädikat `to_int(L,I)`. Es soll den Zahlwert `I` einer Ziffernliste `L` berechnen.

c) Nun definieren Sie ein Prädikat zur Rätsel-Lösung. das man so aufrufen kann:

```
puzzle([S,E,N,D], [M,O,R,E], [M,O,N,E,Y]).
```

Wenn Sie das obige Schema nutzen wollen, müssen Sie eine Liste der unterschiedlichen Variablen berechnen, die in den drei Eingabelisten vorkommen. Benutzen Sie `append` um die Listen zu einer Liste zu konkatenieren, und rufen Sie dann das folgende Prädikat auf, um doppelte Variablen zu eliminieren:

```
distinct([], []).
distinct([E|R], OUT) :-
    (var_member(E, R) -> OUT=L; OUT=[E|L]),
    distinct(R, L).

var_member(X, [Y|_]) :-
    X == Y.
var_member(X, [_|L]) :-
    var_member(X, L).
```

Wenn Sie wollen, können Sie prüfen, ob die folgenden Rätsel lösbar sind, die sich im Internet finden:

- `puzzle([B,I,L,L], [I,R,M,A], [L,I,E,B,E]).`
- `puzzle([A,A,L], [A,A,L], [F,A,N,G]).`
- `puzzle([G,A,U,S,S], [R,I,E,S,E], [E,U,K,L,I,D]).`
- `puzzle([V,A,T,E,R], [M,U,T,T,E,R], [E,L,T,E,R,N]).`

Aufgabe 2 (Hausaufgabe, 3 Punkte)

Zeichnen Sie einen SLD-Baum für die Anfrage `grandfather(ian, G)` an folgendes Programm:

```
 [1] grandfather(X, Z) :- grandparent(X, Z), male(Z).
 [2] grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
 [3] parent(eric, alan).
 [4] parent(eric, bianca).
 [5] parent(fiona, chris).
 [6] parent(fiona, doris).
 [7] parent(ian, eric).
 [8] parent(ian, fiona).
 [9] male(alan).
[10] male(chris).
[11] male(eric).
[12] male(ian).
```

Sie können die Namen gern mit dem ersten Buchstaben abkürzen. Geben Sie den Baum entweder als Bild ab (z.B. auch handgezeichnet und abfotographiert) oder textuell beschrieben. Sie brauchen keine Substitutionen anzugeben (nur die Knoten).

Wenn Sie Ihre Lösung kontrollieren wollen, müssen Sie die Daten in dem folgenden Programm anpassen — siehe Aufgabe 5 unten.

[http://www.informatik.uni-halle.de/~brass/lp24/prolog/sld.pl]

Zur Wiederholung

Aufgabe 3 (Wiederholungs-Fragen)

Was würden Sie in einer mündlichen Prüfung auf die folgenden Fragen zu Bindungsmustern und eingebauten Prädikaten antworten?

a) Definieren Sie den Begriff „Bindungsmuster“ für ein Prädikat `p/n`. Was ist die intuitive Bedeutung?

b) Inwiefern entspricht ein Prädikat zusammen mit einem Bindungsmuster einer klassischen Prozedur? Wie könnte eine Schnittstelle aussehen, mit der man neue eingebaute Prädikate in einem Prolog-System oder einer deduktiven Datenbank nachrüsten könnte?

c) Wann ist ein Bindungsmuster allgemeiner als ein anderes? Wenn man eine Implementierung für ein allgemeineres Bindungsmuster hat, wie kann man damit einen Aufruf mit speziellerem Bindungsmuster ausführen?

d) Welchem Bindungsmuster entspricht ein Tabellenzugriff mit „Full Table Scan“ in einer Datenbank? Z.B. könnte die Anfrage `vater(arno, X)` sein. Wenn man keine Indexstrukturen hat, sondern einfach alle Tupel/Fakten durchgehen muss, wäre die tatsächliche Ausführung wie `vater(V, X), V=arno`. Inwiefern kann man eventuell vorhandene Indexe durch Bindungsmuster beschreiben?

e) Welche eingebauten Prädikate von Prolog haben Sie bei den Hausaufgaben öfters benutzt? Was sind aus Ihrer Sicht die ca. 10 wichtigsten eingebauten Prädikate?

f) Was sind die Unterschiede von `=`, `==`, `=:=` und `is`?

g) Wie können Sie einen beliebigen Term in seine Bestandteile zerlegen (Funktionssymbol und Argumente)?

h) Warum ist `findall` so wichtig? Welche Möglichkeit gibt Ihnen dieses Prädikat, die Sie sonst (nur mit Fakten und Regeln ohne eingebaute Prädikate) nicht hätten? Geben Sie einen Beispiel für einen Aufruf von `findall`.

i) Erläutern Sie die dynamische Datenbank von Prolog. Was sind die wichtigsten eingebauten Prädikate? Warum ist bei modernen Prolog-Systemen eine Deklaration änderbarer Prädikate nötig? Wie schreibt man solche Deklarationen in das Programm?

j) Sie beobachten, dass ein Prädikat, das Sie definiert haben, zunächst die richtige Lösung ausgibt, aber wenn Sie „`;`“ drücken, zu einem „Stack Overflow“ Fehler führt. Was könnte eine mögliche Erklärung für dieses Verhalten sein?

Übungsaufgaben (gemeinsam in der Übung zu bearbeiten)

Aufgabe 4 (Präsenzaufgabe)

Zeichnen Sie einen SLD-Baum für folgenden Programm und die Anfrage `p(X)` (nutzen Sie dabei die "First Literal" Selektionsfunktion wie in Prolog):

```
p(X) :- q1(X), q2(X,c).
p(X) :- q3(X,Y), q4(Y).
q1(X) :- r1(X), r2(X).
q1(X) :- r3(X).
q2(X,Z) :- s1(X), s2(X,Z).
q3(d,e).
q4(e).
r1(a).
r1(b).
r2(b).
r3(c).
s1(b).
s2(b,c).
```

Aufgabe 5 (Präsenzaufgabe)

Angenommen, Regeln sind als "Fakten" (mit Variablen) über ein Prädikat `rule` repräsentiert mit

- dem Kopf der Regel als erstem Argument, und
- der Liste der Rumpf-Literale als zweitem Argument.

Z.B. würde die Regel `p(X) :- q(X), r(X)` so gespeichert:

```
rule(p(X), [q(X), r(X)]).
```

(Normal sind "Fakten" variablenfreie Regeln ohne Rumpf, deswegen ist die Bezeichnung hier nicht ganz korrekt.)
Man kann nun leicht einen Meta-Interpreter in Prolog schreiben, der sämtliche Knoten des SLD-Baums berechnet:

```
sld([]). % Empty Goal: Query proven.
sld([Lit|Rest]) :-
    rule(Lit, Body), % Unification happens here.
    append(Body, Rest, Child),
    sld(Child).
```

Man ruft das Prädikat mit der zu beweisenen Aussage auf, z.B.:

```
sld([p(X)]).
```

Eine erweiterte Version des Programms mit Ausgaben steht in folgender Datei:

[http://www.informatik.uni-halle.de/~brass/lp24/prolog/sld.pl]

In diesem Programm sind die Regeln in der Form `rule(Head, BodyList, VarList)` als Daten repräsentiert, z.B.

```
p(X) :- q1(X), q2(X,c).
```

als

```
rule(p(X), [q1(X), q2(X,c)], [var('X', X)]).
```

Das letzte Argument erlaubt eine verbesserte Ausgabe der Variablen. Nutzen Sie dieses Programm, um sich den SLD-Baum aus e) anzeigen zu lassen.

Aufgabe 6 (Präsenzaufgabe)

Schreiben Sie ein Prädikat `perm(N,L)`, das alle Listen `L`, die sich durch Permutation der Liste `[1,2,...,N]` ergeben, liefert. Z.B. soll `perm(2,L)` die beiden Lösungen `L=[1,2]` und `L=[2,1]` haben. Sie können sich z.B. rekursiv die Permutationen für die Listen bis `N-1` generieren lassen, und dann die Zahl `N` an jeder möglichen Position einfügen (von ganz hinten bis ganz vorne).

Zum Selbstudium

Aufgabe 7 (Zusatzaufgabe)

Schauen Sie sich die folgenden Webseiten an. Sie brauchen für diese Aufgabe nichts abzugeben. Ziel ist, dass Sie einen Eindruck davon gewinnen, was es im Internet zum Thema dieser Vorlesung gibt, und dabei für sich nützliche Quellen entdecken.

- Folien und andere Informationen zur Vorlesung „Deduktive Datenbanken“ von Wolfgang May aus Göttingen gibt es hier:

  [https://www.dbis.informatik.uni-goettingen.de/Teaching/DD-SS21/]

- Stilvorschläge für die Prolog-Programmierung von Michael Covington und anderen finden Sie unter

  [http://www.covingtoninnovations.com/mc/plcoding.pdf]

  U.a. darauf basierend wurden die folgenden Stilempfehlungen entwickelt:

  [https://lifeware.inria.fr/~soliman/post/prolog_guidelines/]

- Einige Dokumente zum ISO-Standard für Prolog stehen unter

  [https://www.deransart.fr/prolog/docs.html]

  Zum Beispiel finden Sie hier die Spezifikation des eingebauten Prädikates `op`:

  [http://www.deransart.fr/prolog/bips.html#operators]