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

20.05.2022

Logische Programmierung & deduktive Datenbanken — Übungsblatt 6 (Ableitung) —

Ihre Lösungen zu den Hausaufgaben a) bis d) geben Sie bitte über die Übungs-Plattform in StudIP ab. Einsendeschluss ist (wegen Himmelfahrt) erst der 02. Juni, 10^{00} (die Aufgaben werden anschließend in der Übung besprochen).

Hausaufgaben

a) Definieren Sie ein Prädikat `ableitung(F,G)`, das mit einem Polynom `F` aufgerufen wird, und `G` an die Ableitung bindet. `F` ist also ein arithmetischer Ausdruck der Form

a_n `*` `x^`n `+ ... +` a_2 `*` `x^2 +` a_1 `*` `x +` a_0

Dabei stehen anstelle der a_i Zahlkonstanten, und natürlich ist auch n eine konkrete Zahl. Ein Beispiel ist:

```
3 * x^2 + 5 * x + 7.
```

Es ist möglich, dass a_i fehlen, also ein Summand die Form `x^`i hat, dann ist $a_i = 1$. Es müssen auch nicht alle Potenzen wirklich auftreten. Das Argument des Polynoms heisst immer `x` (in Prolog ist das ein Atom und keine Variable).

Die Ableitung des Polynoms

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 X + a_0$$

ist

$$a_n * n * x^{n-1} + a_{n-1} * (n-1) * x^{n-2} + \cdots + a_1.$$

Sie müssen also jeweils das Produkt des Exponenten i mit dem Koeffizienten a_i berechnen, und den Exponenten um 1 verringern.

Stellen Sie bitte sicher, dass einfach `x` statt `x^1` gedruckt wird. Ebenso sollte `+0` eliminiert werden. Sie können ggf. ein Hilfsprädikat zum Vereinfachen einer Formel definieren, wenn Sie diese Fälle nicht gleich speziell behandeln wollen.

Testen Sie z.B. die folgenden Fälle:

- `ableitung(x, A)` $\longrightarrow$ `A = 1.`
- `ableitung(27, A)` $\longrightarrow$ `A = 0.`
- `ableitung(x^2 + 5*x + 10, A)` $\longrightarrow$ `A = 2*x+5.`
- `ableitung(2*x^5 + 3*x^2, A)` $\longrightarrow$ `A = 10*x^4+6*x.`

Sie dürfen, wenn Sie möchten, auch weitere Funktionen ableiten. Verlangt sind aber nur Polynome wie oben beschrieben.

b) 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) :- q(X), r(X).
q(a) :- s(Y), t(Y).
q(b).
r(a).
r(b).
s(c).
s(d).
t(c).
```

Sie können den Baum entweder als Bild abgeben (z.B. auch handgezeichnet und abfotographiert) oder textuell beschrieben als Liste von numerierten Knoten, wobei Sie zu jedem Knoten die Liste der Kind-Knoten-Nummern angeben.

c) Definieren Sie eine verallgemeinerte Fakultätsfunktion als Prädikat:

$$f(n,k) := \underbrace{n * (n-1) * \cdots * (n-k+1)}_{k \text{ Faktoren}}$$

Definieren Sie anschliessend ein Prädikat `choose(N, K, Anz)`, das die Anzahl k-elementiger Teilmengen einer n-elementigen Menge ("n über k") berechnet:

$$\binom{n}{k} := \frac{n * (n-1) * \cdots * (n-k+1)}{k * (k-1) * \cdots * 1}$$

d) Definieren Sie ein Prädikat `id_next(N)` zur Generierung eindeutiger Zahlen (mit Hilfe der dynamischen Datenbank). Der erste Aufruf von `id_next(N)` soll `N` an `1` binden, der zweite an `2`, u.s.w. Jeder Aufruf von `id_next(N)` soll nur ein Mal erfolgreich sein, d.h. beim Backtracken sollen keine weiteren Lösungen generiert werden, nur wenn das Prädikat erneut "vorwärts" betreten wird.

Sie speichern sich den aktuellen Zahlwert am besten als Fakt `id_value(N)` in einem dynamischen Prädikat. Dazu müssen Sie `id_value/1` als "`dynamic`" deklarieren. Schreiben Sie sich dann ein Prädikat `id_init`, das eventuell vorhandene Fakten

`id_value(_)` löscht und ein Fakt `id_value(1)` in die dynamische Datenbank speichert. Dieses Prädikat können Sie beim Laden Ihres Programms automatisch aufrufen.

Das Prädikat `id_next(N)` fragt dann den aktuellen Zahlwert mit `id_value(N)` ab, löscht das Fakt, und fügt ein Fakt mit einer um eins größeren Zahl wieder ein.

Zur Wiederholung

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

- Definieren Sie den Begriff "Bindungsmuster" für ein Prädikat `p/n`. Was ist die intuitive Bedeutung?
- 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?
- 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?
- 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?
- Welche eingebauten Prädikate von Prolog haben Sie bei den Hausaufgaben öfters benutzt? Was sind aus Ihrer Sicht die ca. 10 wichtigsten eingebauten Prädikate?
- Was sind die Unterschiede von `=`, `==`, `=:=` und `is`?
- Wie können Sie einen beliebigen Term in seine Bestandteile zerlegen (Funktionssymbol und Argumente)?
- 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`.
- Erläutern Sie die dynamische Datenbank von Prolog. Was sind die wichtigsten eingebauten Prädikate? Warum ist bei modernen Prolog-Systemen eine "`dynamic`"-Deklaration nötig? Wie schreibt man solche Deklarationen in sein Prolog-Programm?
- 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?

f) Was würden Sie in einer mündlichen Prüfung auf die folgenden Fragen zum Cut antworten?

- Was ist das Symbol in Prolog für den Cut?
- Was genau macht der Cut?
- Geben Sie ein kurzes Beispiel für ein logisches Programm, an dem man die Wirkungsweise des Cut erläutern bzw. ausprobieren kann.
- Wozu ist der Cut gut? Erläutern Sie einige typische Anwendungen des Cut. Inwieweit kann er zur Verbesserung der Performance eines Prolog-Programms eingesetzt werden? Was ist eine typische Anwendung für Fallunterscheidungen?
- Warum ist der Cut problematisch?
- Geben Sie ein Beispiel für eine Prädikatdefinition, in der der Cut einen relativ subtilen Fehler verursacht (d.h. einen Fehler, den der Programmierer nicht oder nicht sofort bemerkt, und der deswegen besonders schwierig/gefährlich ist).
- Welche anderen Prolog-Operatoren bzw. Prädikate kann man oft anstelle des Cut verwenden?
- Erläutern Sie das "if-then-else" in Prolog.

Zum Selbstudium

g) 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]