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

27.06.2019

Logische Programmierung & deduktive Datenbanken — Übungsblatt 9 (Eingebaute Prädikate) —

Ihre Lösungen zu den Hausaufgaben g) und h) schicken Sie bitte per EMail an den Dozenten (mit "`[lp19]`" in der Betreff-Zeile). Einsendeschluss ist Mittwoch, der 3. Juli. Aufgabe b) wird dort auch besprochen: Sie sollten vorher darüber nachdenken, müssen aber nichts abgeben.

Zum Selbstudium

a) 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 WWW zum Thema dieser Vorlesung gibt, und dabei für sich nützliche Quellen entdecken.

- Die Symbole für "Modi" von Prädikat-Argumenten ("Eingabe oder Ausgabe") in der SWI-Prolog-Dokumentation sind hier erläutert:

  [http://www.swi-prolog.org/pldoc/man?section=modes]

- Prolog wird auch an Schulen eingesetzt, z.B. gibt es diese Linkliste auf dem Bildungsserver Berlin-Brandenburg:

  [http://bildungsserver.berlin-brandenburg.de/unterricht/faecher/mathematik-naturwissenschaften/informatik/unterrichtsmaterialien-und-fachthemen-ueberblick-und-start/material-inf/prog-sprachen/inf-prolog/]

- Auf dem Hessischen Bildungsserver gibt es eine Seite zu SWI-Prolog, das dort allerdings auch zur Implementierung anderer Sprachen wie Mini-Pascal, Logo und Turtle verwendet wird:

  [http://arbeitsplattform.bildung.hessen.de/fach/informatik/swiprolog/]

- Webseite eines Lehrers zu Prolog-Kursen an der Schule:

  [https://www.tinohempel.de/info/info/prolog/index.htm]

  Das zitierte "Arbeitsbuch Prolog" von Hartmut Göhner und Bernd Hafenbrak findet sich z.B. hier:

  [http://www.erasmus-reinhold-gymnasium.de/info/prolog/arbeitsbuch_prolog.pdf]

b) Was würden Sie in einer mündlichen Prüfung auf die folgenden Fragen 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?

Präsenzaufgaben

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 `sum(X,Y,Z)`, das genau dann gilt, wenn `X + Y = Z`, und das die Bindungsmuster `bbf`, `bfb`, `fbb` und `bbb` behandeln kann. Sie können die verschiedenen Fälle unterscheiden mit `var(X)`, `nonvar(X)` bzw. `number(X)`.

e) 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.

f) Definieren Sie ein Prädikat `calc`, das eine Eingabeaufforderung ("Prompt") ausdruckt, einen arithmetischen Ausdruck einliest (ohne Variablen), den Wert dieses Ausdrucks berechnet, und das Ergebnis ausgibt. Dies soll in einer Schleife wiederholt werden, bis der Benutzer "`quit`" eingibt. Sie können voraussetzen, dass der Nutzer jede Zeile mit "`.`" abschliesst. Syntaxfehler brauchen Sie nicht zu behandeln.

Hausaufgabe

Die folgenden Aufgaben setzen das "Vier gewinnt" Spiel vom letzten Aufgabenblatt fort.

g) Schreiben Sie ein Prädikat `waehle_zug(Zustand, Farbe, Spalte)`, das in einem gegebenen Spielzustand `Zustand` für eine gegebene Farbe einen sinnvollen Zug liefert, d.h. eine Spalte (von `1` bis `7`) auswählt.

Es darf dabei keine Spalte gewählt werden, die schon voll ist, d.h. bei der in Zeile `6` schon ein Stein steht (Zeilen seinen hier von unten gezählt, so dass der erste Stein in Zeile `1` landet). Sollte das Spielbrett vollständig voll sein, muss `waehle_zug` scheitern.

Außerdem ist eine Mindestanforderung, dass, wenn ein Gewinn in einem Zug möglich ist, auch in die entsprechende Spalte gesetzt wird. Ansonsten können Sie sich aussuchen, wie "clever" Ihr Programm ist. Sie könnten z.B. noch einen Zug vorausschauen, ob der Gegener dann gewinnen wird, und solche Züge soweit noch möglich vermeiden. Wenn Sie wollen, informieren Sie sich z.B. über den min-max-Algorithmus:

[https://de.wikipedia.org/wiki/Minimax-Algorithmus]

Achten Sie aber darauf, dass Sie nicht zu viel Rechenzeit verbrauchen. Es soll ein Zug nach spätestens ca. 5 Sekunden geliefert werden.

Nutzen Sie für den Zugriff auf den Spielzustand möglichst nur die im letzten Blatt definierten Prädikate. So wäre es möglich, auch zwei Programme gegeneinander spielen zu lassen, weil sie beide den Spielzustand verstehen können. Hierfür wäre es auch nützlich, wenn die Namen Ihrer Prädikate zur Zugberechnung z.B. Ihre Initialen als Suffix enthalten (um Namenskonflike zu vermeiden).

h) Schreiben Sie nun noch ein einfaches Hauptprogramm, das den Spieler gegen Ihr Programm aus g) spielen läßt. Der Mensch fängt an mit den gelben Steinen. Es soll dann jeweils ein Zug abgefragt werden, und ausgegeben werden, in welche Spalte der Computer setzt. Außerdem soll das Spielfeld sowohl nach dem Zug des Menschen als auch nach dem Zug des Computers angezeigt werden. Schließlich soll nach jedem Zug getestet werden, ob einer der beiden Spieler gewonnen hat, oder ob es zu einem Unentschieden kommt, weil das Spielfeld voll ist. Es muss dann eine entsprechende Ausgabe erfolgen und keine weiteren Züge abgefragt werden.

Wahrscheinlich gibt es auch zu dieser Aufgabe fertige Lösungen im Internet. Ich denke aber, dass es für den Lerneffekt besser ist, das Programm selbst zu schreiben. Mindestens müssen Sie offenlegen, welche Quelle Sie verwendet haben, und zusätzliche Kommentare einfügen und die Schnittstelle so anpassen, dass die Spezifikation aus der Aufgabenstellung genau erfüllt wird. Von einem möglichen Turnier wären Sie ausgeschlossen, wenn Sie das Programm nicht selbst geschrieben haben.