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

06.04.2017

Logische Programmierung & deduktive Datenbanken — Übungsblatt 1 (Einführung) —

Ihre Lösungen zu den Hausaufgaben h) bis k) schicken Sie bitte per EMail an den Dozenten (mit "`[lp17]`" in der Betreff-Zeile). Einsendeschluss ist der 17. April (die Aufgaben werden in den Übungen am 18. April und 20. April besprochen). 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 nicht alles zu verstehen, sondern sich nur einen ersten Überblick verschaffen, was es gibt. Es ist geplant, dass auf jedem Übungsblatt einige Quellen genannt werden, die für Sie interessant sein könnten.

- "Prolog" in der Wikipedia:

  [https://de.wikipedia.org/wiki/Prolog_(Programmiersprache)]

- "Learn Prolog Now!":

  [http://www.learnprolognow.org/].

  Schauen Sie sich insbesondere Abschnitt 1.1 "Some Simple Examples" der "Free Online Version" an. Es gibt auch eine Version mit Auswertungsmöglichkeit:

  [http://lpn.swi-prolog.org/lpnpage.php?pagetype=html&pageid=lpn-htmlse1]

- Es empfiehlt sich, ein Prolog-System auf dem eigenen Rechner zu installieren. Es gibt viele freie Prolog-Systeme, die sich mehr oder weniger an den Prolog-Standard halten. Ein bekanntes und problemloses System ist:

  [http://www.swi-prolog.org/]

  Im Laufe des Semesters werden wir uns auch andere Prolog-Systeme anschauen, die vielleicht eine höhere Performance oder mehr Möglichkeiten im Bereich "Constraint Logic Programming" bieten. Aber für den Einstieg ist SWI-Prolog sicher eine gute Option. Falls Sie auf Ihrem Rechner nichts installieren wollen, können Sie auch die Online-Version testen:

  [http://swish.swi-prolog.org/]

Sie brauchen für diese Aufgabe nichts abzugeben, aber sollten schon einige Zeit investieren (es sind ja zwei Stunden pro Woche für Hausaufgaben eingeplant, zusätzlich noch Zeit für Selbstudium).

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

- Was bedeutet eigentlich deklarative Programmierung? Warum ist SQL eine deklarative Sprache?
- Welche Vorteile hat deklarative Programmierung? (Sie könnten das z.B. anhand von SQL-Anfragen erläutern, die Sie mit entsprechenden Java-Programmen vergleichen. Vielleicht erinnern Sie sich an die ersten Übungsaufgaben in der Vorlesung "Datenbanken I", bei denen dieser Vergleich praktisch ausprobiert wurde.)
- Welche logischen Operatoren können in Prolog-Regeln verwendet werden (soweit bisher an Beispielen in der Vorlesung gezeigt)? Wie sind Regeln aufgebaut? (Zum Vergleich könnten Sie den Bereichskalkül für das relationale Modell aus "Datenbanken I" heranziehen.)
- Wie kann man den Inhalt einer relationalen Datenbank in Prolog darstellen?
- Wie unterscheiden sich Variablen und Konstanten in Prolog syntaktisch?
- Nennen Sie ein Prolog-System.
- Wie kann man in Prolog eine Datei mit Fakten und Regeln laden?
- Wie kann man ein Prolog-System verlassen, d.h. die Kommandoschleife des Interpreters beenden? (Es heisst nicht "`quit`".)

Präsenzaufgaben

c) Laden Sie sich die folgende Datei mit den Beispiel-Fakten zu Verwandtschaftsbeziehungen herunter:

[http://users.informatik.uni-halle.de/~brass/lp17/family.pl]

Diese Datei enthält Fakten zu folgenden Prädikaten, die im ersten Beispiel der Vorlesung genutzt werden:

- `parents(Child, Father, Mother)`.
- `couple(Husband, Wife)`.
- `man(Man)`.
- `woman(Woman)`.

Starten Sie ein Prolog-System. (Auf unseren Linux-Rechnern öffnen Sie dazu ein Terminal-Fenster und geben dann "`prolog`" ein.)

Nach dem Start zeigen die meisten Prolog-Interpreter mit "`?-`" an, dass sie auf eine Anfrage (Beweisziel) warten. Auch Befehle, die den Zustand des Prolog-Interpreters verändern, werden als Beweisziele behandelt.

Laden Sie nun die Datei "`family.pl`". Dies sollte mit einem der folgenden Befehle funktionieren:

- Falls Sie das Prolog-System im gleichen Verzeichnis gestartet haben:

  ```
  ['family.pl'].
  ```

  Vergessen Sie den Punkt "`.`" am Ende nicht, der in Prolog Fakten, Regeln und Beweisziele abschließt.

- Falls die Datei im aktuellen Verzeichnis steht und das Prolog-System mit der Default-Endung "`.pl`" konfiguriert wurde:

  ```
  [family].
  ```

  Probieren Sie bei Bedarf, ob es funktioniert, wenn Sie die Datei in "`family.pro`" umbenennen: "`.pl`" ist die klassische Endung für Prolog, leider aber auch für Perl. Man kann bei der Installation von SWI-Prolog auch die Endung "`.pro`" wählen. Auch diese Endung wird aber teils von anderen Anwendungen genutzt.

- Wenn Sie das Prolog-System unter Windows gestartet haben, ist sein aktuelles Verzeichnis das Installationsverzeichnis. Dann müssen Sie den vollen Pfad eingeben:

  ```
  ['C:/sb/lp17/family.pl'].
  ```

  Bei SWI-Prolog können Sie sich das aktuelle Verzeichnis mit folgender Anfrage anzeigen lassen (nicht im Prolog-Standard enthalten):

  ```
  working_directory(X, X).
  ```

  Das Prädikat hat zwei Argumente, weil man das aktuelle Verzeichnis damit auch ändern kann.

- Da die eckigen Klammern nur eine Abkürzung für den Aufruf des System-Prädikates "`consult`" sind, geht es auch so:

  ```
  consult('family.pl').
  ```

  Dies zeigt nochmal deutlich, dass es syntaktisch eine Anfrage ist, die ein vordefiniertes Prädikat aufruft.

Es ist kein Problem, wenn Sie die Datei mehrfach laden. Die vorherige Definition der Prädikate wird jeweils gelöscht. Es ist ganz typisch, dass Sie einen Editor in einem Fenster offen haben, dort das Programm entwicklen und jeweils nur abspeichern, und im anderen Fenster ein Prolog-System haben, und dort die veränderte Datei jeweils neu laden.

Falls Sie die Definition eines Prädikates zunächst aus Datei A geladen haben, und dann aus einer anderen Datei B laden, bekommen Sie eine Warnung ("Redefined static procedure"). Auch in diesem Fall wird die frühere Definition also gelöscht (die Standard-Eingabe "`user`" zählt dabei jedes Mal als neue Datei).

d) Die Relation `parents` entspricht folgender Tabelle:

parents		
Child	Father	Mother
emil	arno	birgit
frida	chris	doris
gerd	chris	doris
ian	emil	frida
julia	emil	frida
klaus	gerd	helga

Probieren Sie die folgenden Anfragen aus:

- `parents(frida, X, Y).`
  Wer sind die Eltern von Frida?

- `parents(X, _, frida).`
  Wer sind die Kinder von Frida? (Für die anonyme Variable "`_`" wird keine Antwort ausgegeben, sie dient nur dazu, die hier nicht benötigte "Vater"-Spalte zu füllen.)

Prolog unterscheidet sich von relationalen Datenbanken u.a. dadurch, dass jeweils nur eine Antwort auf einmal berechnet wird. Wenn es weitere Antworten geben könnte, bleibt der Cursor nach der Ausgabe am Ende der Zeile stehen und das System wartet auf Ihre Entscheidung:

- Sie können "Enter"/"Return" drücken, dann ist die Abfrage beendet und es werden keine weiteren Antworten gesucht.

- Drücken Sie dagegen "`;`" ("oder?"), so wird nach einer alternativen Lösung (Antwort) gesucht. Es ist möglich, dass keine gefunden wird. In diesem Fall wird "`false`" ausgegeben (oder "`no`").

Wenn das Prolog-System dagegen schon weiß, dass es keine weitere Antwort gibt, beendet es die Abfrage und zeigt wieder die Eingabeaufforderung "`?-`". (Dann sollten Sie nicht "`;`" eingeben, das wäre ein Syntaxfehler.)

e) Probieren Sie auch einige ja/nein-Abfragen und lassen Sie sich die ganze Relation `parents` ausgeben. (D.h. lassen Sie sich die ganze Extension des Prädikats "`parents`" drucken.)

f) Schreiben Sie die folgenden Prädikatdefinition (aus der Vorlesung) in eine Datei und laden Sie diese Datei:

```
father(X,Y)        :- parents(X, Y, _).
mother(X,Y)        :- parents(X, _, Y).
parent(X, Y)       :- father(X, Y).
parent(X, Y)       :- mother(X, Y).
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
person(X, m)       :- man(X).
person(X, f)       :- woman(X).
```

Editoren unter Linux sind z.B. `gedit`, `gvim`, `emacs` (wenn Sie keinen kennen, verwenden Sie `gedit`).

Stellen Sie einige Anfragen, um die Defitionen zu testen.

Sie können auch unterschiedliche Formatierungen testen, um zu sehen, wie Sie die Regeln als besonders übersichtlich empfinden. Z.B. kann man jede atomare Formel auf eine eigene Zeile schreiben, und die Formeln im Rumpf (rechte Seite der Regel) einrücken. Sie können auch Kommentare einfügen (von "`%`" bis zum Ende der Zeile).

Probieren Sie auch die Reaktion auf mindestens einen Syntaxfehler aus.

g) Es soll nun eine Endlosschleife (Endlos-Rekursion) getestet werden. Definieren Sie dazu folgende Regel:

```
p :- p.
```

Sie müssen diese Regel nicht in eine Datei schreiben, sondern können mit

```
[user].
```

auch von der Standard-Eingabe lesen (beachten Sie die Eingabe-Aufforderung "`|:`"). Die Eingabe wird beendet mit `Ctrl-D` (Steuerung + D). So definierte Regeln gehen verloren, wenn das Prolog-System beendet wird (das ist in diesem Fall aber kein Problem, sondern sogar erwünscht). Stellen Sie nun die Anfrage

```
p.
```

Dazu müssen Sie den "Definitionsmodus" verlassen, und die Eingabeaufforderung muss wieder "`?-`" sein.

Bei Prolog sind Endlosschleifen möglich (im Gegensatz zu deduktiven Datenbanken). Die Ausführung dieser Anfrage endet nicht (freiwillig). Drücken Sie "`Ctrl+C`", um die Anfrage abzubrechen. Sie kommen dadurch in einen Unterbrechungsmodus, der verschiedene Möglichkeiten bietet, den Debugger zu nutzen. "`h`" zeigt verschiedene Optionen. Zunächst brauchen Sie nur "`a`" (für "abort"), um die Ausführung der Anfrage zu beenden.

Hausaufgabe

h) Die gegebene Datenbank enthält auch eine Tabelle `couple` für Ehepaare:

couple	
Husband	Wife
arno	birgit
chris	doris
emil	frida
gerd	helga
klaus	lena

Definieren Sie ein Prädikat "`married_with`" mit der gleichen Information, das aber symmetrisch ist: Wenn X mit Y verheiratet ist, ist natürlich auch Y mit X verheiratet. Terminieren Anfragen an Ihr Prädikat? Versuchen Sie ggf. das Problem zu lösen.

i) Definieren Sie ein Prädikat "`siblings`" für Geschwister (d.h. `siblings(X,Y)` soll gelten, wenn `X` und `Y` den gleichen Vater und die gleiche Mutter haben). Sie können die Bedingung `X \= Y` im Rumpf verwenden, um `X` $\neq$ `Y` zu fordern.

j) Definieren Sie ein Prädikat "`uncle`": `uncle(X,Y)` soll gelten, wenn `Y` ein Bruder eines Elternteils von `X` ist.

k) Definieren Sie in Prädikat `inconsistent`, dass bei Integritätsverletzungen herleitbar ist: Die Information über das Geschlecht einer Person ist ja teilweise redundant gespeichert. Sie können selbstverständlich auch Hilfsprädikate definieren, mit denen Sie dann `inconsistent` herleiten können.

Prolog erlaubt das Überladen von Prädikaten mit unterschiedlicher Stelligkeit (Anzahl von Argumenten): Sie können ein Prädikat `inconsistent(X)` definieren, das Personen `X` enthält, die nach der Datenbank gleichzeitig männlich und weiblich sein müssen. Dieses Prädikat können Sie dann benutzen, um `inconsistent` ohne Argumente herzuleiten, wenn es mindestens eine solche Integritätsverletzung gibt. Beachten Sie, dass man bei Prolog für Prädikate ohne Argumente auch keine Klammern "`()`" schreibt.

Der Beispiel-Zustand enthält keine Fehler. Wenn Sie in einem `parents`-Fakt Vater und Mutter vertauschen, sollte aber `inconsistent` herleitbar sein, weil das Geschlecht jetzt nicht mehr mit den Angaben aus `man` und `woman` übereinstimmt. Ebenso, wenn Sie Mann und Frau in einem `couple`-Fakt vertauschen.

Die Aufgabe ist erfüllt, wenn das Prädikat inkonsistente Geschlechts-Angaben findet. Man kann aber auch Schlüssel prüfen, dass z.B. eine Person nicht zwei verschiedene Väter bzw. Mütter hat. Ebenso, dass ein Mann nicht gleichzeitig mit zwei Frauen verheiratet ist, und umgekehrt. Für Schlüssel brauchen Sie wieder das Prädikat `X \= Y`.

Sie können auch Fremdschlüssel prüfen: Alle Väter und alle Ehemänner sollten in der Tabelle `man` verzeichnet sein, und entsprechen für Frauen. Hierfür benötigen Sie die Negation: Mit `\+ man(X)` können Sie testen, dass `X` nicht in der Tabelle `man` eingetragen ist. Dabei muss `X` schon weiter links an einen Wert gebunden sein.