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

19.06.2014

Logische Programmierung & deduktive Datenbanken — Hausaufgabe 2 —

Schreiben Sie einen einfachen Typ-Prüfer für Prolog/Datalog-Regeln in Prolog. Zur vereinfachung sei angenommen, dass die Prädikate der zu prüfenden Regeln nur ganze Zahlen und Atome als Argumente haben, keine strukturierten Terme. Die Prädikate, die in diesen Regeln verwendet werden dürfen, seien im Typprüfer selbst deklariert, und zwar in folgender Form:

```
% komponist(KNR, NAME, VORNAME, GEBOREN, GESTORBEN)
pred(komponist(int, atom, atom, int, int)).

% stueck(SNR, KNR→komponist, TITEL, TONART, OPUS).
pred(stueck(int, int, atom, atom, atom)).

pred(<(int, int)).
pred(ergebnis(int, int)).
...
```

D.h. die verwendbaren Prädikate sind (vorläufig) in den Typprüfer fest eingebaut. Programmieren Sie ein Prädikat `check/1`, so dass man auf folgende Art die Typprüfung durchführen kann:

```
read(X), check(X).
```

Die Eingabe `X` kann dabei eine Regel, ein einzelnes Literal, oder eine Konjunktion von Literalen (Anfrage, Beweisziel) sein. Die syntaktische Richtigkeit können Sie voraussetzen. Das eingebaute Prädikat `read` liefert die Regel/Anfrage auch bereits als Term (Baumstruktur). Wenn Sie `:-` und `,` als normale Funktoren verwenden wollen, müssen Sie diese Symbole in `'...'` einschließen, z.B. `':-'`.

Es reicht aus, wenn Ihr Prädikat `check` fehlschlägt, falls die Eingabe einen Typfehler enthält. Man bekommt in diesem Fall also die Antwort "no" und sonst "yes". Falls Sie wollen, können Sie natürlich auch bessere Fehlermeldungen ausgeben (mit `write` und `nl`), bevor Ihr Prädikat fehlschlägt (ggf. hierzu `fail` verwenden). Gute Fehlermeldungen verkomplizieren die Aufgabe allerdings deutlich.

Zum Beispiel sollte die Eingabe

```
ergebnis(X, Y) :- komponist(_, 'Wolfgang Amadeus', 'Mozart', X, Y).
```

als korrekt gewertet werden, die folgenden aber alle nicht:

- `ergebnis(X, Y) :- komponist(X, Y, Y, 'Wolfgang Amadeus', 'Mozart').`
  Atom-Konstanten an Integer-Argumentpositionen.
- `ergebnis(X1, X2) :- komponist(Nr, X1, X2, Y1, Y2).`
  `X1` und `X2` müssten nach dem Rumpf-Literal Atome sein, aber nach dem Kopf-Literal ganze Zahlen.
- `ergebnis(X, X) :- komponist(Nr, 'Wolfgang Amadeus', 'Mozart', X).`
  Es gibt kein Prädikat `komponist` mit nur vier Argumenten.

Die Bereichsbeschränkung brauchen Sie nicht zu prüfen, d.h. es soll legal sein, Variablen im Kopf zu verwenden, die im Rumpf nicht vorkommen.

Sie können diese Aufgabe in drei Schritten lösen:

- Überführen Sie die Eingabe in eine Liste von Literalen.
- Ersetzen Sie die Konstanten in diesen Literalen durch die Typbezeichner `atom` und `int`. Den Test auf die Typen können Sie mit den eingebauten Prädikaten `atom/1`, `integer/1` und `var/1` durchführen. Um Literale in Prädikat und Argumentliste zu zerlegen, können Sie das Prädikat `=..` verwenden. Damit können Sie auch umgekehrt wieder Literale erzeugen.
- Nun müssen Sie nur noch jedes Literal Ihrer Liste mit den gegebenen `pred`-Fakten unifizieren. Dafür ist wichtig, dass Sie im zweiten Schritt die Variablen aus der Eingabe erhalten, so dass eine inkonsistente Verwendung der gleichen Variable mit unterschiedlichen Typen auch bemerkt wird.

Zusatzaufgabe: Lesen Sie die `pred`-Fakten und die zu prüfenden Regeln aus einer Datei. Definieren Sie also ein Prädikat `check_file/1`, dem man den Namen der zu prüfenden Datei mit Prädikat-Deklarationen und Programm-Regeln übergibt.

Termin: Diese Aufgaben werden in der Übung am 26.06.2014 besprochen.