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

20.06.2024

Logische Programmierung & deduktive Datenbanken — Übungsblatt 10 (Minimales Modell) —

Ihre Lösungen zu den Hausaufgaben 1 bis 4 geben Sie bitte über die Übungs-Plattform in StudIP ab. Einsendeschluss ist Mittwoch, der 26. Juni, 18^{00} (die Aufgaben werden in der Übung am Donnerstag besprochen).

Hausaufgaben

(Die Hausaufgaben dieses Blattes stammen aus der Klausur von 2012.)

Aufgabe 1 (Hausaufgabe, 4 Punkte)

Schreiben Sie ein Prädikat `vereinfachen(F, E)`, das eine aussagenlogische Formel `F` nach den unten angegebenen Regeln in eine einfachere Formel `E` überführt. Formeln sind dabei als Terme mit den zweistelligen Funktionssymbolen `and` und `or` repräsentiert. Die aussagenlogischen Konstanten in den Formeln sind als Prolog-Atome dargestellt. Es sei nun angenommen, dass eine Analyse ergeben hat, dass einige der aussagenlogischen Konstanten falsch sein müssen, diese wurden in der Formel bereits durch die Zahl `0` ersetzt. D.h. Formeln können neben Atomen auch die Zahl `0` enthalten, die für `false` steht. Die Vereinfachung soll durch Auswertung des Wertes `0` nach folgenden Regeln erfolgen:

- `and(F,0)` und `and(0,F)` werden in `0` überführt.
- `or(F,0)` und `or(0,F)` werden in `F` überführt.

Selbstverständlich soll dies auch rekursiv für alle Bestandteile der Formel geschehen, z.B. muss `and(f,and(0,g))` zu `0` vereinfacht werden (d.h. man muss die Teilformeln zuerst vereinfachen, bevor man die obigen Regeln anwendet). Sie können nur die im Skript genannten eingebauten Prädikate verwenden. Wahrscheinlich brauchen Sie das einstellige Prädikat `atom(T)`, das genau dann wahr ist, wenn `T` ein Atom ist (daher ist z.B. `atom(0)` falsch). Sie können sich selbstverständlich beliebige Hilfsprädikate in Prolog definieren. Sie können davon ausgehen, dass das erste Argument eine Termstruktur wie oben beschrieben ist.

Aufgabe 2 (Hausaufgabe, 2 Punkte)

Beschreiben Sie bitte mit eigenen Worten, was das folgende Prolog-Prädikat tut:

```
p([a,a|_]).
p([_|R]) :- p(R).
```

Geben Sie dabei auch mindestens einen Beispiel-Aufruf an, der erfolgreich ist.

Aufgabe 3 (Hausaufgabe, 2 Punkte)

Was gibt das folgende Programm aus, wenn man das Prädikat `main` aufruft? (Versuchen Sie das Ergebnis herauszufinden, ohne das Programm tatsächlich auszuführen — wenn Sie wollen, können Sie das hinterher zur Kontrolle tun.)

```
main :- p(b), t, write('!').

p(a) :- !, write(1), q(a).
p(b) :- write(2), r(b), !, q(b).
p(X) :- write(3), s(Y), !, q(Y).
p(X) :- write(4), !, q(b).

q(X) :- write(X).

r(a).

s(c).
s(d).

t :- write(5), u(X), write(X), fail.
t.

u(e).
u(f).
```

Aufgabe 4 (Hausaufgabe, 2 Punkte)

Sind die folgenden beiden Literale unifizierbar? Falls ja: Geben Sie einen allgemeinsten Unifikator an. Falls nein: Begründen Sie Ihre Antwort kurz (welche Variablenbindungen werden ausgeführt und warum geht es dann nicht weiter?).

- p(X, f(X), X, g(f(a)))
- p(Y, Z, a, g(Z))

Zur Wiederholung

Aufgabe 5 (Wiederholungs-Fragen)

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

a) Nennen Sie Unterschiede von Datalog zu Prolog.

b) Nennen Sie einige Varianten von Datalog, d.h. welche Konstrukte könnte man zu Basis-Datalog noch hinzutun? (Ein oder zwei Möglichkeiten reichen.)

c) Warum denken Prolog-Programmierer und Datalog-Programmierer anders über definite Hornklauseln?

d) Welche Regeln sind „allowed“? Geben Sie zunächst die erste/einfachste Variante der Definition wieder. Warum ist diese Eigenschaft für die praktische Bottom-Up-Auswertung mit dem T_P-Operator wichtig? Geben Sie ein Beispiel für eine Regel, die nicht „allowed“ ist, und erläutern Sie, welches Problem es bei der Berechnung des minimalen Modells geben würde.

   Hinweis: Beachten Sie aber, dass der Satz über den Zusammenhang des minimalen Modells mit dem kleinsten Fixpunkt des T_P-Operators ohne die Einschränkung auf „allowed“-Regeln gilt. Diese Eigenschaft ist nur für die praktische Berechnung durch Anwendung des Satzes wichtig.

e) Wenn man auch eingebaute Prädikate im Regelrumpf erlaubt, z.B. „<“, warum ist die einfache Definition zulässiger Regeln („allowed“, s.o.) nicht ausreichend? Was wäre eine einfache Korrektur? Warum ist diese Lösung doch sehr eingeschränkt? Würde es für einfache SQL-Anfragen ausreichen?

f) Für die entgültige Definition zulässiger Regeln („range-restricted rule given a binding pattern β for the head“) haben wir angenommen, dass eine Menge erlaubter Bindungsmuster $valid(p)$ für jedes Prädikat p definiert sind. Was ist dann die Intuition der Definition einer bereichsbeschränkten Regel? Warum braucht man für die Bottom-Up-Auswertung mit dem T_P-Operator „stark bereichsbeschränkte“ Regeln? Was ist da der Unterschied?

g) Warum braucht man in Datalog keine volle Unifikation, sondern nur einfaches „Matching“? Was ist der Unterschied?

h) Wie kann man Funktionssymbole von Prolog mit eingebauten Prädikaten in einer deduktiven Datenbank simulieren? Erläutern Sie das am Beispiel von Listen. Was wäre die wesentliche Einschränkung gegenüber Prolog? Wenn man in der deduktiven Datenbankbank auch eine funktionale Notation erlauben würde, was könnte man damit gegenüber Prolog gewinnen?

Übungsaufgaben (gemeinsam in der Übung zu bearbeiten)

Aufgabe 6 (Präsenzaufgabe)

Mit den folgenden Regeln kann man den T_P-operator implementieren:

```
derivable(Head, Facts) :-
    rule(Head, BodyList, _Vars),
    is_true(BodyList, Facts).

is_true([], _).
is_true([Lit|BodyRest], Facts) :-
    member(Lit, Facts),
    is_true(BodyRest, Facts).

tp(FactsIn, FactsOut) :-
    findall(Fact, derivable(Fact, FactsIn), FactsOut).
```

Die Repräsentation der Regeln ist wie auf dem Blatt 8 für die SLD-Resolution, nämlich in der Form `rule(Head, BodyList, VarList)`. Z.B. wird die Regel

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

folgendermaßen als Daten gespeichert:

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

Das letzte Argument erlaubt eine verbesserte Ausgabe der Variablen, wird aber hier nicht benötigt (wenn Sie nicht bei der Regelanwendung die Substitution ausgeben wollen). Sie dürfen das Argument auch entfernen.
Machen Sie sich ein einfaches Beispiel und prüfen Sie, dass diese Regeln den T_P-Operator korrekt implementieren. Sie finden obige Regeln (und die für die folgende Teilaufgabe) unter

[http://www.informatik.uni-halle.de/~brass/lp24/prolog/tp0.pl]

Aufgabe 7 (Präsenzaufgabe)

Die Iteration des T_P-Operators können Sie mit den folgenden Regeln in Prolog definieren:

```
minmod(Facts) :-
    minmod_iter(0, [], Facts).

minmod_iter(N, Facts, MinMod) :-
    tp(Facts, NewFacts),
    (has_changed(Facts, NewFacts) ->
        NextN is N+1,
        minmod_iter(NextN, NewFacts, MinMod);
        MinMod = Facts).

has_changed(Facts, NewFacts) :-
    member(Fact, NewFacts),
    \+ member(Fact, Facts).
```

Bauen Sie Ausgaben in dieses Programm ein, so dass man die iterative Berechnung des minimalen Modells mit dem T_P-Operator gut nachvollziehen kann.
Machen Sie ein kleines Beispiel, das mindestens drei Iterationen benötigt.

Zum Selbstudium

Aufgabe 8 (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.

a) Schauen Sie sich das XSB System an:

[http://xsb.sourceforge.net/]

Das XSB System ist ein Prolog System mit Tabellierung, so dass die Terminierung für Datalog-Programme garantiert ist. Den Artikel „XSB as an Efficient Deductive Database Engine“ von Konstantinos Sagonas, Terrance Swift, und David S. Warren (SIGMOD'94, 442–453) finden Sie hier:

[https://doi.org/10.1145/191839.191927]

Falls Sie nicht im Netz der Universität sind, und daher kostenlosen Zugriff auf die Artikel der ACM haben, gibt es hier eine kostenlos zugreifbare Version:

[https://www.researchgate.net/publication/
2792472_XSB_as_an_Efficient_Deductive_Database_Engine]

b) Zu der Vorlesung “Knowledge Representation” von Marek Sergot gibt es Vorlesungsmaterialien hier:

[https://www.doc.ic.ac.uk/~mjs/teaching/491.html]

Sie könnten sich z.B. das Kapitel “Minimal models and fixpoint semantics for definite programs” anschauen:

[https://www.doc.ic.ac.uk/~mjs/teaching/
KnowledgeRep491/Fixpoint_Definite_491-2x1.pdf]

c) Sie finden meine Habilitationsschrift “Bottom-Up Query Evaluation in Extended Deductive Databases” (Universität Hannover, 1997) z.B. hier:

[http://nbn-resolving.de/urn:nbn:de:gbv:089-2367669109]

Zumindest die Kapitel 1 und 2 wären auch für diese Vorlesung sehr lesenswert.