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

27.06.2024

Logische Programmierung & deduktive Datenbanken — Übungsblatt 11 (Bottom-Up-Auswertung) —

Ihre Lösungen zu den Hausaufgaben 1 bis 4 geben Sie bitte über die Übungs-Plattform in StudIP ab. Einsendeschluss ist Mittwoch, der 3. Juli, 18^{00} (die Aufgaben werden in der Übung am Donnerstag besprochen).

Hausaufgaben

(Diese Aufgaben stammen aus der Klausur von 2012.)

Aufgabe 1 (Hausaufgabe, 3 Punkte)

Ein Silvester-Feuerwerker hat Daten der Feuerwerksartikel folgendermaßen als Fakten repräsentiert:

```
% artikel(ID, Name, Typ, Brenndauer).
artikel(1, 'Überraschungsvulkan', vulkan, 20).
artikel(2, 'Schweizer Supervulkan', vulkan, 45).
artikel(3, 'Silbersonne', sonne, 90).
artikel(4, 'Tanz der Vampire', batterie, 25).
artikel(5, 'Diamant-Rakete', rakete, 4).
...
```

Außerdem hat er seine Erfahrungen mit den Artikeln bei Silvester-Feuerwerken in Form von Bewertungen abgespeichert (das erste Argument ist ein Fremdschlüssel):

```
% bewertung(ID, Jahr, Note, Text).
bewertung(1, 2011, 1, 'toll!').
bewertung(1, 2012, 2, 'hatte ich größer in Erinnerung').
bewertung(2, 2012, 1, 'sehr hübsch: Gold mit Blinkersternen').
bewertung(3, 2009, 1, 'dreht sich problemlos, brennt lange').
bewertung(4, 2010, 1, 'rote Blinker und Gold').
bewertung(5, 2010, 3, 'Na ja').
...
```

Schreiben Sie bitte eine Anfrage in Datalog, die ID und Name aller Bodenfeuerwerksartikel (`vulkan` oder `sonne`) liefert, die mindestens 30s brennen, und mindestens eine Bewertung mit der Note `1` haben (daneben kann es beliebige weitere Bewertungen geben). Das Ergebnis soll im Prädikat `anfrage` stehen. Sie können beliebige Hilfsprädikate definieren.

Aufgabe 2 (Hausaufgabe, 3 Punkte)

Zeichen Sie den Prädikat-Abhängigkeitsgraphen für das folgende Datalog-Programm P (die Argumente sind für diese Aufgabe nicht wichtig, daher wurde ein aussagenlogisches Programm gewählt):

```
a ← b ∧ c.
b ← a.
c ← d ∧ e.
d ← f.
e ← f.
f.
```

Zeichnen Sie außerdem den reduzierten Prädikat-Abhängigkeitsgraphen.

Aufgabe 3 (Hausaufgabe, 3 Punkte)

Gegeben sei noch einmal das Programm P aus Aufgabe 2. Berechnen Sie das minimale Modell von P durch Iteration des T_P-Operators. Geben Sie die Mengen $I_0 := \emptyset$, $I_i := T_P(I_{i-1})$ für $i = 1, 2, \ldots$ an, bis ein Fixpunkt erreicht ist. Selbstverständlich brauchen Sie nur die jeweils neuen Fakten explizit anzugeben, z.B. in der Form $I_2 = I_1 \cup \{\ldots\}$.

Aufgabe 4 (Hausaufgabe, 3 Punkte)

Gegeben sei folgendes logische Programm:

```
p(X,Y)  ← q(X,1) ∧ t(Y).    % (1)
q(X,Y)  ← r(X,Y) ∧ s(X,Z). % (2)
r(a,1).                     % (3)
r(b,2).                     % (4)
s(a,3).                     % (5)
s(b,4).                     % (6)
t(c).                       % (7)
t(d).                       % (8)
```

Geben Sie den (vollständigen) SLD-Baum für folgende Anfrage an:

```
p(A,B).
```

Schreiben Sie an jede Kante die Nummer der angewendeten Regel (bzw. des Faktes). Die Substitutionen in jedem Schritt brauchen Sie nicht anzugeben, aber die berechneten Antwortsubstitutionen schon.

Aufgabe 5 (Hausaufgabe, 0 Punkte)

Falls Sie am „Vier Gewinnt“ Turnier teilnehmen wollen, laden Sie bitte ihre aktuelle Programmversion als Aufgabe 5 in die Übungsplattform hoch. Diese kann gerne gegenüber Ihrer früheren Abgabe nochmals verbessert sein.

Zur Wiederholung

Aufgabe 6 (Wiederholungs-Fragen)

Was würden Sie in einer mündlichen Prüfung auf die folgenden Fragen zur Bottom-Up Auswertung antworten?

a) Was ist das Ziel der Bottom-Up Auswertung? Was ist die theoretische Grundlage?

b) Erläutern Sie den Prädikat-Abhängigkeitsgraphen. Wie ist er definiert? Wozu ist er gut?

c) Definieren Sie „rekursives Prädikat“ und „rekursive Regel“.

d) Wie bestimmt man eine Auswertungsreihenfolge für die Prädikate?

e) Was ist der “reduzierte Prädikat-Abhängigkeitsgraph”? Welche Bedeutung hat das Ergebnis einer topologischen Sortierung dieses Graphen?

f) Wie kann man ein relationales DBMS zur Bottom-Up Auswertung nutzen? Beschreiben Sie insbesondere auch, wie dabei Datalog-Regeln in SQL-Befehle übersetzt werden.

g) Erklären Sie, wie man Operationen der relationalen Algebra zur Auswertung von Regeln verwenden kann.

h) Wie kann man die Auswertung von Datalog-Regeln selbst implementieren (in einer Sprache wie C++ oder Java)?

i) Diskutieren Sie die Bedeutung der Duplikat-Eliminierung.

Aufgabe 7 (Wiederholungs-Fragen)

Was würden Sie in einer mündlichen Prüfung auf die folgenden Fragen zur seminaiven Auswertung antworten?

a) Was ist der Zweck der “seminaiven Auswertung”?

b) Erläutern Sie die Idee der seminaiven Auswertung in dem einfachen Fall, dass es nur ein rekursives Rumpfliteral gibt.

c) Beschreiben Sie die seminaive Auswertung genauer: Welche vier Varianten eines Prädikats (unterschiedliche Tupelmengen) werden durch die seminaive Auswertung eingeführt? Wie werden die Regeln umgeschrieben? Welche Befehle werden zur Initialisierung vor der Schleife und zum Weiterschalten nach jedem Schleifendurchlauf ausgeführt?

d) Betrachten Sie den Fall einer Regel mit zwei rekursiven Rumpfliteralen. Durch die seminaive Auswertung spart man von den vier möglichen Kombinationen nur eine, nämlich den Join-Anteil von alten mit alten Tupeln. Warum ist das trotzdem wichtig und lohnt den Aufwand?

e) Warum braucht man bei einer Regel mit n rekursiven Rumpfliteralen nicht $2^n - 1$ Versionen der Regel, sondern nur n?

Übungsaufgaben (gemeinsam in der Übung zu bearbeiten)

Aufgabe 8 (Präsenzaufgabe)

Berechnen Sie

a) den Prädikat-Abhängigkeitsgraphen und

b) den reduzierten Prädikat-Abhängigkeitsgraphen

des folgenden Programms (die q_i sind EDB-Prädikate):

$$
\begin{array}{lcl}
\mathsf{p}_1 & \leftarrow & \mathsf{q}_1 \wedge \mathsf{q}_2. \\
\mathsf{p}_1 & \leftarrow & \mathsf{q}_1 \wedge \mathsf{q}_3. \\
\mathsf{p}_2 & \leftarrow & \mathsf{p}_1. \\
\mathsf{p}_3 & \leftarrow & \mathsf{q}_3. \\
\mathsf{p}_3 & \leftarrow & \mathsf{p}_1. \\
\mathsf{p}_3 & \leftarrow & \mathsf{p}_2. \\
\mathsf{p}_4 & \leftarrow & \mathsf{p}_2 \wedge \mathsf{p}_3. \\
\mathsf{p}_5 & \leftarrow & \mathsf{p}_4 \wedge \mathsf{q}_2.
\end{array}
\qquad
\begin{array}{lcl}
\mathsf{p}_6 & \leftarrow & \mathsf{p}_5. \\
\mathsf{p}_4 & \leftarrow & \mathsf{p}_6. \\
\mathsf{p}_7 & \leftarrow & \mathsf{p}_5 \wedge \mathsf{p}_3. \\
\mathsf{p}_7 & \leftarrow & \mathsf{q}_4 \wedge \mathsf{p}_7. \\
\mathsf{p}_8 & \leftarrow & \mathsf{p}_2. \\
\mathsf{p}_9 & \leftarrow & \mathsf{p}_8 \wedge \mathsf{p}_7.
\end{array}
$$

Beantworten Sie aber folgende Fragen:

c) Welche der Prädikate sind rekursiv?

d) Welche Regeln sind rekursiv?

e) Was sind die rekursiven Cliquen?

f) Geben Sie eine Berechnungsreihenfolge für die Regeln des Programms an. Markieren Sie, welche Regeln in Schleifen iteriert werden müssen.

Zum Selbstudium

Aufgabe 9 (Zusatzaufgabe)

In dieser Aufgabe sollen Sie ein ganz kleines Textadventure-Spiel erstellen.

Textadventure-Spiele sind eine Art Bücher, bei denen der Leser/Spieler den Ablauf der Handlung beeinflußt. Es wird ihm/ihr jeweils die aktuelle Situation beschrieben, z.B.:

> „Du bist auf einem Waldweg, der hier eine Biegung macht. Im Norden befindet sich eine Lichtung, während im Osten der Weg tiefer in den Wald hineinführt. Du hörst das Summen vieler Bienen."

Der Spieler kann nun Kommandos/Spielzüge eingeben, wie z.B. „(ich) gehe nach Norden". Tatsächlich wird dieses Kommando meist durch „n" abgekürzt (das ist für den Spieler wie den Programmierer bequemer). Natürlich kann der Spieler nur in Richtungen gehen, die der Programmierer/Autor vorgesehen hat: Auf das Kommando „s" müsste in der obigen Situation geantwortet werden „dort ist der Wald zu dicht" (oder einfach „das geht nicht"). So ein Spiel ist häufig auf eine recht kleine Zahl von Orten/Räumen beschränkt, an denen sich der Spieler befinden kann (kommerzielle Spiele aus den Achtzigern etwa 100).

Das bloße Herumgehen auf der vom Programmierer entworfenen Karte und das Entdecken neuer Orte würde aber schnell langweilig. Ein anderes wichtiges Kommando ist es, Sachen zu untersuchen oder näher zu betrachten. Dadurch bekommt der Spieler häufig zusätzliche Informationen oder findet versteckte Gegenstände. Im Beispiel könnte er durch „Betrachte Bienen" ein Astloch mit Honigwaben finden. Schließlich ist es noch wichtig, Gegenstände nehmen zu können, z.B. „Nimm Honig". Natürlich würden die Bienen das verhindern.

Hier gilt es also eine echte Aufgabe zu lösen, wie man nun doch an den Honig herankommt. Zum Beispiel könnte sich auf der Lichtung eine Hütte befinden, und in dieser vielleicht eine Pfeife, deren Rauch die Bienen abschreckt. Der Spieler muss sich also in einer bestimmten Reihenfolge bestimmte Gegenstände verschaffen und dabei eine gewisse Phantasie entwickeln. Solche Gegenstände können ihm dann auch neue Wege öffnen. So würde der Honig vielleicht einen Bären besänftigen, der den Eingang zu einer Höhle versperrt. In der Höhle befindet sich z.B. ein Schatz, der das Ziel des Spieles darstellt.

Sie können sich natürlich auch eine andere Geschichte ausdenken. Ihr Spiel sollte mindestens 3 „Räume" und einen Gegenstand haben, und neben den Kommandos für die vier Himmelsrichtungen auch noch mindestens zwei weitere Kommandos („Verben") „verstehen" (z.B. „nimm Gegenstand", und „betrachte Gegenstand"). Es ist nicht verlangt, dass Ihr Spiel eine sinnvolle Geschichte enthält — es reicht, wenn man in der kleinen Spielwelt herumgehen kann und den Gegenstand nehmen und betrachten kann.

Sie können natürlich die Grammatik für Textadventure-Befehle aus Kapitel 7 ausprobieren und ggf. etwas erweitern oder modifizieren. Sie müssen dazu auch den Scanner am Ende des Kapitels programmieren, um Eingabezeichen bis zum Zeilenende einzulesen und daraus Worte zu machen (Prolog Atome), die dann von der Grammatik verarbeitet werden können. Alternativ können Sie mit der Grammatik auch bis auf die Zeichen-Ebene herunter gehen.

Die Ausgaben für die Kommandos dürfen recht einfach sein, aber vielleicht haben Sie ja auch Spass daran, eine einfache Spielwelt zu implementieren. Versuchen Sie, Ihr Programm möglichst übersichtlich zu strukturieren.

Aufgabe 10 (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) Herr Wenzel und ich sind dabei, Datalog-basierte Sprachen zur Programmierung von Microcontrollern (z.B. Arduino) zu entwickeln. Wir sind uns bezüglich der genauen Sprache noch nicht ganz einig, aber „Konkurrenz belebt das Geschäft“. Man muss wohl auch verschiedene Möglichkeiten anschauen, um zu einer guten Sprache zu kommen. Sie sind natürlich eingeladen, da mitzuwirken.

   Die Webseite von Herrn Wenzel ist:

   [https://dbs.informatik.uni-halle.de/microlog/]

   Insbesondere haben wir auch eine Implementierung vorgeschlagen, die auf erweiterten endlichen Automaten basiert.

b) Meine Sprache ist leicht anders. In dem Artikel ging es aber in erster Linie nicht um die Sprache, sondern um die Verifikation von Invarianten/Integritätsbedingungen:

   [https://users.informatik.uni-halle.de/~brass/micrologS/]

c) In meinem Artikel auf der RuleML+RR 2021 habe ich eine Sprache vorgeschlagen, die auf Events basiert:

   [https://users.informatik.uni-halle.de/~brass/micrologE/rr21.pdf]

   Die Folien meines Vortrags finden Sie hier:

   [https://users.informatik.uni-halle.de/~brass/micrologE/rr21talk.pdf]

   Leider ist die Implementierung noch „under Construction“. Ich hoffe, dass ich in diesem Sommer dazu kommen werde. Auch hierzu wäre natürlich Mithilfe willkommen.