Logische Programmierung & deduktive Datenbanken — Übungsblatt 4 (Terme und Listen) —

Ihre Lösungen zu den Hausaufgaben (Aufgabe 1 bis 4) geben Sie bitte über die Übungsplattform in StudIP ab. Einsendeschluss ist Mittwoch, der 15. Mai, 16⁰⁰ (die Aufgaben werden in der Übung am Donnerstag besprochen).

Hausaufgaben

Aufgabe 1 (Hausaufgabe, 2 Punkte)

Definieren Sie ein Prädikat `ablist(L)`, das wahr ist, wenn `L` eine Liste nur aus den Elementen `a` und `b` ist. Z.B. soll `ablist([a,a,b,a])` wahr sein, aber `ablist([a,b,c,a])` falsch sein. Es ist nur gefordert, dass keine anderen Elemente außer `a` und `b` vorkommen, nicht, dass diese beiden enthalten sein müssen. Z.B. soll `ablist([])` wahr sein.

Aufgabe 2 (Hausaufgabe, 2 Punkte)

Schreiben Sie ein Prädikat `sum(L, S)`, das die Zahlen in einer Liste `L` aufsummiert. Z.B. soll der Aufruf `sum([1,2,3], S)` die Lösung `S = 6` liefern. Mit `number(E)` können Sie testen, ob ein Element `E` der Liste eine Zahl ist. Ihr Prädikat soll fehlschlagen (`false` liefern), wenn ein Element der Liste keine Zahl ist. Für die leere Liste soll die Summe `0` geliefert werden.

Aufgabe 3 (Hausaufgabe, 2 Punkte)

Schreiben Sie ein Prädikat `insert(E, LIn, LOut)`, das ein Element `E` in die sortierte Liste `LIn` an die richtige Stelle einfügt. Z.B. soll `insert(3, [1,4,5], X)` das Ergebnis `X = [1,3,4,5]` liefern. Sie können davon ausgehen, dass `LIn` nur Zahlen enthält und `E` eine Zahl ist. Falls `E` schon in `LIn` vorkommt, soll es nicht nochmals eingefügt werden.

Aufgabe 4 (Hausaufgabe, 2 Punkte)

Schreiben Sie ein Prädikat `mysort(LIn, LOut)`, das die Liste `LIn` sortiert, indem es die Elemente mittels `insert` aus Aufgabe 3 nacheinander in eine anfangs leere Liste einfügt. Duplikate werden dabei eliminiert.

Zur Wiederholung

Aufgabe 5 (Wiederholungs-Fragen)

Was würden Sie in einer mündlichen Prüfung auf die folgenden Fragen zur Prolog-Syntax und zu Listen antworten?

- a) Was wird `write(+ (1,2))` ausgeben?
- b) Was wird `display(1+2)` ausgeben?
- c) Wird die Anfrage `1+1 = 2` mit `true` oder `false` beantwortet?
- d) Mit welchem Funktionssymbol und welcher Konstanten werden Listen in Prolog intern repräsentiert?
- e) Geben Sie drei verschiedene Schreibweisen für die Liste „`[1,2,3]`“ in Prolog an.
- f) In welchem Prolog-Term wird Notation `[F|R]` intern übersetzt? Wofür stehen `F` und `R` hier?
- g) Wie wird das Prädikat `append` in Prolog definiert? Geben Sie die Regeln an.
- h) Funktioniert `append` auch zum Aufspalten einer Liste?
- i) Definieren Sie ein Prädikat `member(X, L)` zum Test, ob `X` ein Element der Liste `L` ist.
- j) Definieren Sie ein Prädikat `length(L, N)`, das die Länge `N` einer Liste `L` berechnet.
- k) Was muss man in Prolog beachten, damit rekursiv definierte Prädikate terminieren?
- l) Wie sehen Regeln in Prolog aus? Definieren Sie die Begriffe „Regel“, „Kopf einer Regel“, „Rumpf einer Regel“, „Kopfliteral einer Regel“, „Rumpfliteral einer Regel“, „Regel über ein Prädikat `p`“.

Aufgabe 6 (Wiederholungs-Fragen)

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

- a) Definieren Sie den Begriff „Substitution“.
- b) Was ist das Ergebnis der Anwendung der Substitution $\{X/Y, Y/a\}$ angewendet auf $p(X, Y, Z, b)$?
- c) Definieren Sie den Begriff „Unifikator“ und „Allgemeinster Unifikator“ („most general unifier“, mgu).
- d) Ist der allgemeinste Unifikator zweier Literale immer eindeutig bestimmt?

-
- e) Was ist ein allgemeinsten Unifikator von $p(X, a)$ und $p(Y, Z)$? Was wäre ein Beispiel für einen nicht allgemeinsten Unifikator?
 - f) Berechnen Sie einen allgemeinsten Unifikator von $p(X, X)$ und $p(f(Y), f(a))$.
 - g) Wie spielen Unifikatoren für Prolog eine Rolle? Wo werden sie gebraucht? Wie kann man sie mit Prolog berechnen?
 - h) Was unterscheidet die Unifikation von Zuweisungen in klassischen Programmiersprachen?
 - i) Was ist der „Occur Check“? Warum ist er für die Unifikation wichtig? Welches Problem gibt es? Warum lassen viele Prolog-Systeme ihn weg? Gibt es noch alternative Lösungen?
 - j) Wie kann man testen, ob ein Prolog-System den Occur-Check verwendet? Geben Sie ein einfaches Beispiel-Programm an.

Übungsaufgaben (gemeinsam in der Übung zu bearbeiten)

Aufgabe 7 (Präsenzaufgabe)

Definieren Sie ein Prädikat mit folgender Regel:

```
p(L*R) :-  
    write('L = '), write(L), nl,  
    write('R = '), write(R), nl,  
    nl.
```

[https://users.informatik.uni-halle.de/~brass/lp24/prolog/p_mult.pl]

Neu ist, dass das Kopfliteral der Regel einen strukturierten Term als Argument hat. Probieren Sie folgende Anfragen aus:

- a) `p(3*4)`.
Beachten Sie, dass nicht das Ergebnis „12“ an das Prädikat übergeben wird, sondern der Term „3*4“ (also die Baumstruktur).
- b) `p(*(3,4))`.
Das ist die gleiche Anfrage mit dem Term in der Standardnotation „Funktionssymbol(Argumentterme)“.
- c) `p(1*2*3)`.
Die implizite Klammerung ist von links, weil `*` den Typ `yfx` hat.
- d) `p(1*(2*3))`.
Man beachte, dass die Klammern nur der Eingabe des Terms dienen. Der rechte Teilterm braucht bei der Ausgabe keine Klammern mehr, und wird dann auch ohne Klammern ausgegeben.
- e) `p(1*((2*3)*4))`.
Hier kann man nochmal sehen, dass bei der Ausgabe nur die nötigen Klammern gedruckt werden. Intern ist einfach die Baumstruktur des Terms gespeichert, und es gibt keine Information über Klammern mehr.
- f) `p(2+3)`.
Da dieser Term nicht `*` als Funktor hat, scheitert die „Unifikation“, mit der Anfrage und Regelkopf in Übereinstimmung gebracht werden sollen. Da es keine weitere Regel über `p` gibt, wird auch die Anfrage mit „false“ beantwortet.
- g) `p(1+2*3)`.
Hier ebenso: Der Term enthält zwar ein „*“, aber wegen der Prioritäten („Punktrechnung vor Strichrechnung“) ist der Operator in der Wurzel der Baumstruktur das „+“. Deswegen können der Term aus der Anfrage und der Term in Kopf der Regel nicht durch Variablenbindungen in Übereinstimmung gebracht werden.

h) $p(2)$.

i) $p(X)$.

Dieses Beispiel zeigt, dass die Unifikation in beiden Richtungen funktioniert: Die Variable X kann mit dem Term $L * R$ in Übereinstimmung gebracht werden, indem X an $L * R$ gebunden wird. In der Ausgabe erscheinen interne Repräsentationen der Variablen L und R (vermutlich sind die Zahlen die Hauptspeicheradressen).

Aufgabe 8 (Präsenzaufgabe)

Definieren Sie ein Prädikat $\text{ableitung}(F,G)$, das mit einem Polynom F aufgerufen wird, und G an die Ableitung bindet. F ist also ein arithmetischer Ausdruck der Form

$$a_n * x^n + \dots + a_2 * x^2 + a_1 * x + a_0$$

Dabei stehen anstelle der a_i Zahlkonstanten, und natürlich ist auch n eine konkrete Zahl. Ein Beispiel ist:

$$3 * x^2 + 5 * x + 7.$$

Es ist möglich, dass a_i fehlen, also ein Summand die Form x^i hat, dann ist $a_i = 1$. Es müssen auch nicht alle Potenzen wirklich auftreten. Das Argument des Polynoms heisst immer x (in Prolog ist das ein Atom und keine Variable).

Die Ableitung des Polynoms

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

ist

$$a_n * n * x^{n-1} + a_{n-1} * (n-1) * x^{n-2} + \dots + a_1.$$

Sie müssen also jeweils das Produkt des Exponenten i mit dem Koeffizienten a_i berechnen, und den Exponenten um 1 verringern.

Stellen Sie bitte sicher, dass einfach x statt x^1 gedruckt wird. Ebenso sollte $+0$ eliminiert werden. Sie können ggf. ein Hilfsprädikat zum Vereinfachen einer Formel definieren, wenn Sie diese Fälle nicht gleich speziell behandeln wollen.

Testen Sie z.B. die folgenden Fälle:

- $\text{ableitung}(x, A) \rightarrow A = 1$.
- $\text{ableitung}(27, A) \rightarrow A = 0$.
- $\text{ableitung}(x^2 + 5*x + 10, A) \rightarrow A = 2*x+5$.
- $\text{ableitung}(2*x^5 + 3*x^2, A) \rightarrow A = 10*x^4+6*x$.

Sie können in der Übung auch in Gruppen zusammenarbeiten. Wenn Sie eine Aufteilung in ein simples Ableitungsprädikat und ein Vereinfachungsprädikat wählen, wäre auch möglich, dass jeder in der Gruppe nur an der einen Hälfte arbeitet. Bei Bedarf bin ich gern bereit, zu helfen.

Zum Selbststudium

Aufgabe 9 (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) Tutorial zu Prolog von John R. Fisher „prolog :- tutorial“:

[<https://saksagan.ceng.metu.edu.tr/courses/ceng242/documents/prolog/jrfisher/contents.html>]

Es gibt auch eine Version mit Inhaltsverzeichnis und Inhalt in getrennten Frames:

https://saksagan.ceng.metu.edu.tr/.../prolog/jrfisher/pt_framer.html

- b) Eine Online-Version des Buches „Adventure in Prolog“ von Dennis Merritt gibt es hier:

[<http://www.amzi.com/AdventureInProlog/apreface.php>]

Das Buch ist 1990 im Springer-Verlag erschienen. Es ist ein Prolog-Tutorial, das ein Textadventure-Spiel als durchgehendes Beispiel verwendet. Die Folien zu einem Prolog-Kurs von Dennis Merritt finden Sie hier:

[https://github.com/AmziLS/prolog_course/blob/master/prolog_course.pptx]

- c) Es gibt dort auch ein ehemals kommerzielles Prolog, das jetzt Open Source ist:

[<http://www.amzi.com/index.php>]

Videos zur Einbindung von Amzi-Prolog in die Eclipse-Entwicklungsumgebung finden Sie hier:

[<http://www.amzi.com/videos/>]

- d) Ein Eclipse-Plugin für SWI-Prolog (PDT 3.1 von der Uni Bonn) gibt es hier:

[<https://sewiki.iai.uni-bonn.de/research/pdt/docs/start>]

Die Installation ist in diesem Video beschrieben:

[<https://www.youtube.com/watch?v=5LlsH9MxnWs>]

- e) Wenn Sie Video-Tutorials mögen, könnten Sie z.B. das folgende Tutorial zur Listen-Notation von „EducationAboutStuff“ probieren (etwas langatmig):

[<https://www.youtube.com/watch?v=Jrf6CKc2c0k>]

Hier ist ein anderes Tutorial (von „mycurlycode“):

[<https://www.youtube.com/watch?v=WKoM4L74XzY>]

Das folgende Video (von „bSimple“) enthält eine Lösung zu Aufgabe b):

[<https://www.youtube.com/watch?v=PUyccUj405k>]