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

05. Februar 2004

Deduktive Datenbanken & Logische Programmierung — Klausur —

Name: ______________________________

Matrikelnummer: ______________________________

Studiengang: ______________________________

Aufgabe	Punkte	Max. Punkte	Zeit
1 (Prolog Programm)		10	25 min
2 (Minimales Modell)		6	15 min
3 (SLD Resolution)		6	15 min
4 (Prolog Syntax)		3	10 min
5 (Unifikation)		1	5 min
Summe		26	70 min

Hinweise:

- Bearbeitungsdauer: 90 Minuten
- Skript, Bücher, Notizen sind erlaubt. Notebooks, PDAs, etc. dürfen nicht verwendet werden. Mobiltelefone bitte ausschalten.
- Die Klausur hat 8 Seiten. Bitte prüfen Sie die Vollständigkeit.
- Bitte benutzen Sie den vorgegebenen Platz. Wenn Sie auf die Rückseite ausweichen müssen, markieren Sie bitte klar, daß es eine Fortsetzung gibt.
- Tauschen Sie keinesfalls irgendwelche Dinge mit den Nachbarn aus. Notfalls rufen Sie eine Aufsichtsperson zur Kontrolle.
- In dem Prolog-Programm gibt es auch für kleine Syntaxfehler kleine Punkt-Abzüge.
- Fragen Sie, wenn Ihnen eine Aufgabe nicht klar ist!

Aufgabe 1 (Prolog Programm) 10 Punkte

Man kann Relationen (wie in relationalen Datenbanken) in Prolog z.B. als Liste von Listen darstellen: Jede innere Liste entspricht einem Tupel, die äußere Liste dann der Menge von Tupeln. So würde etwa die Liste

```
[[a,p], [b,q], [c,q], [d,r]]
```

der folgenden Relation entsprechen:

$1	$2
a	p
b	q
c	q
d	r

Zur Vereinfachung nehmen wir an, daß die Attribute/Spalten hier durch ihre Position (1, 2, ...) identifiziert werden, und nicht über Namen. Natürlich soll Ihr Programm mit beliebigen Relationen funktionieren, und nicht nur den obigen Beispieldaten. Insbesondere können Sie keine Annahmen über die maximale Spaltenanzahl treffen.

Ihre Aufgabe besteht nun darin, eine einfache Variante der Selektion in der relationalen Algebra zu implementieren. Das Prädikat `select(R, A, V, S)` soll für eine gegebene Relation `R`, eine Attributposition `A`, und einen Wert `V` die Menge der Tupel aus `R` liefern, die an Position `A` den Wert `V` haben. Diese Teilmenge von `R` soll an die Variable `S` gebunden werden (und nicht etwa mit `write` ausgegeben werden). Sie können davon ausgehen, daß das Prädikat nur mit dem Bindungsmuster `bbbf` aufgerufen wird. Zum Beispiel sollte

```
select([[a,p], [b,q], [c,q], [d,r]], 2, q, S)
```

die Variable `S` an die Liste `[[b,q], [c,q]]` binden. Der Beispiel-Aufruf entspricht dann der Selektion $\sigma_{\$2='q'}(\mathtt{R})$.

Sie können nur die im Skript genannten eingebauten Prädikate verwenden, z.B. `is`, `>`, `=`, `\=`, `\+`.

Sie können sich selbstverständlich beliebige Hilfsprädikate in Prolog definieren. Es könnte z.B. geschickt sein, sich zuerst ein Prädikat `nth_elem(L, N, E)` zu definieren, das das `N`-te Element `E` aus der Liste `L` auswählt.

Sie können davon ausgehen, daß die Eingabe korrekt ist, es also wirklich ein Attribut mit der gewünschten Position gibt, und daß alle Tupel gleich lang sind.

Es ist Platz für die Lösung auf der nächsten Seite der Klausur.

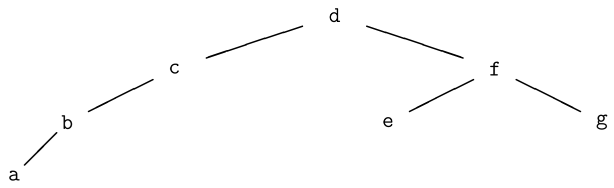
Definieren Sie `select(R, A, V, S)`:

- `R`: Eingaberelation
- `A`: Attribut-Position (1, 2, ...)
- `V`: Wert
- `S`: Ausgaberelation

Aufgabe 2 (Minimales Modell, T_P-Operator) 6 Punkte

In einer deduktiven Datenbank kann man binaäre Bäume z.B. mit den drei Relationen `degree`, `left_child` und `right_child` darstellen. Dabei gibt das Prädikat `degree` die Anzahl der Kindknoten an: Bei `degree(N,0)` ist `N` ein Blattknoten, bei `degree(N,1)` hat `N` nur einen linken Kind-Knoten, und bei `degree(N,2)` hat `N` sowohl einen linken als auch einen rechten Kind-Knoten. Man könnte `degree` auch aus den Daten über `left_child` und `right_child` berechnen, aber dazu wäre die nichtmonotone Negation nötig.

Zum Beispiel wäre der Baum



durch folgende Fakten beschrieben:

```
degree(a, 0).
degree(b, 1).
degree(c, 1).
degree(d, 2).
degree(e, 0).
degree(f, 2).
degree(g, 0).
left_child(b, a).
left_child(c, b).
left_child(d, c).
left_child(f, e).
right_child(d, f).
right_child(f, g).
```

Das folgende logische Programm soll die Höhe eines Teilbaums berechnen:

```
height(N, 0)   :-  degree(N, 0).
height(N, H)   :-  degree(N, 1), left_child(N, C), height(C,H).
height(N, H1)  :-  degree(N, 2), left_child(N, L), height(L,H1),
                   right_child(N, R), height(R,H2), H1 >= H2.
height(N, H2)  :-  degree(N, 2), left_child(N, L), height(L,H1),
                   right_child(N, R), height(R,H2), H2 >= H1.
```

Sei P dieses logische Programm zusammen mit den Fakten für den Beispiel-Baum. 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 $I_2 = I_1 \cup \{\ldots\}$. Die Menge der Fakten für den Beispiel-Baum heiße F.

Aufgabe 3 (SLD Baum) 6 Punkte

Aussagenlogische Ausdrücke mit `and` ($\wedge$) und `or` ($\vee$) kann man in Prolog als Terme darstellen, etwa steht `and(p, or(q,r))` für $p \wedge (q \vee r)$. Wenn für jedes aussagenlogische Symbol der Wahrheitswert bekannt ist, kann man z.B. auf folgende Art prüfen, ob eine Formel erfüllt ist:

```
(1)  is_true(X) :- prop(X, true).
(2)  is_true(and(X,Y)) :- is_true(X), is_true(Y).
(3)  is_true(or(X,Y)) :- is_true(X).
(4)  is_true(or(X,Y)) :- is_true(Y).
(5)  prop(p, true).
(6)  prop(q, false).
(7)  prop(r, true).
```

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

```
is_true(and(p, or(q,r)))
```

Schreiben Sie an jede Kante die Nummer der angewendeten Regel. Die Substitutionen brauchen Sie nicht anzugeben.

Aufgabe 4 (Prolog Syntax) 3 Punkte

Was gibt `read(Rule), display(Rule)` aus, wenn der Benutzer folgendes eingibt:

```
p([a]) :- q, r, s -> t; u.
```

Das eingebaute Prädikat `display` gibt den Term in Standard-Syntax aus. Alternativ dürfen Sie auch den Operatorbaum für den Term zeichnen. Die relevanten Operator-Deklarationen sind:

Operator	Typ	Priorität
:-	xfx	1200
;	xfy	1100
->	xfy	1050
,	xfy	1000

Aufgabe 5 (Unifikation) 1 Punkt

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