BLATT 7

DANIEL SCHMIDT & PAMELA FLEISCHMANN

Aufgabe 1. Sei das folgende Datalog-Programm gegeben:

$$gn(X,Y) :- gl(X,Y).$$
$$gn(X,Y) :- kp(X,X1), gn(Y1,X1), kp(Y,Y1).$$

und die Zielklausel $? - gn(c,Y)$.
Schritt 1: Es gilt die Regel $query^f(Y) :- gn^{bf}(c,Y).$ einzufügen, womit das komplette Programm wie folgt lautet:

$$r_0 = query^f(Y) :- gn^{bf}(c,Y).$$
$$r_1 = gn^{bb}(X,Y) :- gl^{ff}(X,Y).$$
$$r_2 = gn^{bb}(X,Y) :- kp^{ff}(X,X1), gn^{ff}(Y1,X1), kp^{ff}(Y,Y1).$$

Schritt 2: Als nächstes müssen alle Vorkommen von IDB-Prädikaten im Rumpf verändert werden, in diesem Fall ist r_2, sodass sich folgende Regeln ergeben:

$$magic\_r_0\_gn^{bf} :- .$$
$$magic\_r_2\_gn^{bb} :- kp^{ff}(X,X1), \quad magic\_r_2\_gn^{bb}(X1,Y1), kp^{ff}(Y,Y1).$$

Aufgabe 2.

Aufgabe 3.

Aufgabe 4. ad a.
Die Ausgaben des Original Programms sind wie folgt:

```
ready>>?notintoys(susan).
CORAL::Warning : Using underbound method 0 for notintoys 1!
(Number of Answers = 0)
ready>>?notintoys(john).
CORAL::Warning : Using underbound method 0 for notintoys 1!
yes.
        ... next answer ? (y/n/all)[y] all
(Number of Answers = 1)
```

Folgene Anpassungen an das Programm waren notwendig:

```
module flounder_example.
export notintoys(f).
```

```
person(john).
person(susan).

employed(susan,marketing).

notintoys(X) :- person(X), Y = toys, not employed(X,Y).

end_module.
```

Die resultierende Ausgabe ist die folgende:

```
ready>>consult(flounder.P).
ready>>?notintoys(susan).
CORAL::Warning : Using underbound method 0 for notintoys 1!
yes.
        ... next answer ? (y/n/all)[y] all
(Number of Answers = 1)
```

ad b.

Das Programm sieht wie folgt aus:

```
module serie.
export zshg(bf,bb).

zshg(G, Y) :- allNodes(G, LA), allConnectedNodes(G,LC), sameLength(LA, LC

sameLength(LA, LC) :- length(LA, LLA), length(LC, LLC), equal(LLA, LLC).

connected(K1, K2) :- equal(K1, K2).
connected(K1, K2) :- kante(K1, K2).
connected(K1, K2) :- kante(K2, K1).

path(K1, K2) :- connected(K1, K2).
path(K1, K3) :- connected(K1, K2), connected(K2, K3), not equal(K1, K3),

allNodes(G, []).
allNodes(G, [H|T]) :- node(G, H), not member(H, T).

reachableNodes(G, X, []).
reachableNodes(G, X, [H|T]) :- node(G, H), connected(X, H), not member(H,

canReachAllNodes(G, X) :-

allConnectedNodes(G, []).
allConnectedNodes(G, [H|T]) :- node(G, H), not member(H, T), reachableNod

end_module.
```

Die folgende Ausgabe wird erzeugt:

log