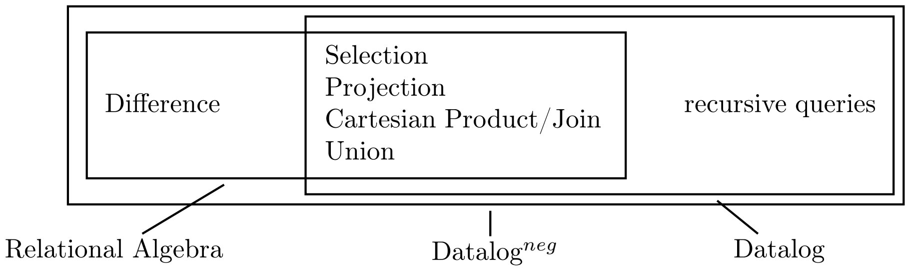
AG Datenbanken und Informationssysteme · Institut für Informatik · Universität Göttingen

Database Theory
Winter Term 2013/14
Prof. Dr. W. May

4. Unit: Datalog

Discussion by 15./22.1.2014

Exercise 1 (Äquivalenz von Algebra und Datalog) Show that for every expression of the relational algebra there is an equivalent stratified Datalog program.



Union Let p, q relations. Then, for $u = p \cup q$

```
u(X1,...,XN) :- p(X1,...,XN).
u(X1,...,XN) :- q(X1,...,XN).
```

Difference Let p, q relations. Then, for $d = p \setminus q$

```
d(X1,...,XN) :- p(X1,...,XN), not q(X1,...,XN).
```

Projection Let p a relation with attributes $X_1, \ldots, X_n$. Then, for $pr = \pi[X_{i_1}, \ldots, X_{i_k}](p)$ with $X_{i_j} \in \{X_1, \ldots, X_n\}$

```
pr(XI1,...,XIK) :- p(X1,...,XN).
```

Selection Let p a relation with attributes $X_1, \ldots, X_n$, α a condition over $X_1, \ldots, X_n$. Then, for $s = \sigma[\alpha](p)$

s(X1,...,XN) :- p(X1,...,XN), α.

Join Let p, q relations with common attributes $X_k, \ldots, X_m$.
Then, for $j = p \bowtie q$

```
j(X1,...,XN) :- p(X1,...,XK,...,XM), q(XK,...,XM,...,XN).
```

The program that corresponds to a complex algebra expression is stratified since each subexpression defines a new predicate symbol, and thus the dependency graph corresponds to the tree structure of the expression.

Exercise 2 (Datalog to Algebra)
Consider the translation of Datalog programs with a distinguished answer predicate to the relational algebra.

- Given a rule $B \leftarrow C_1 \wedge \ldots \wedge C_m \wedge \neg D_{m+1} \wedge \ldots \wedge \neg D_{m+n}$
  where the C_i and D_i are of the form $R_i(a_1, \ldots, a_\ell)$, a_j constants or variables. Give an algebra expression that returns the relation defined by it.
- Which additional construct must also be translated?
- Consider the following program (arbitrary arity of predicates, each rule assumed to be safe):

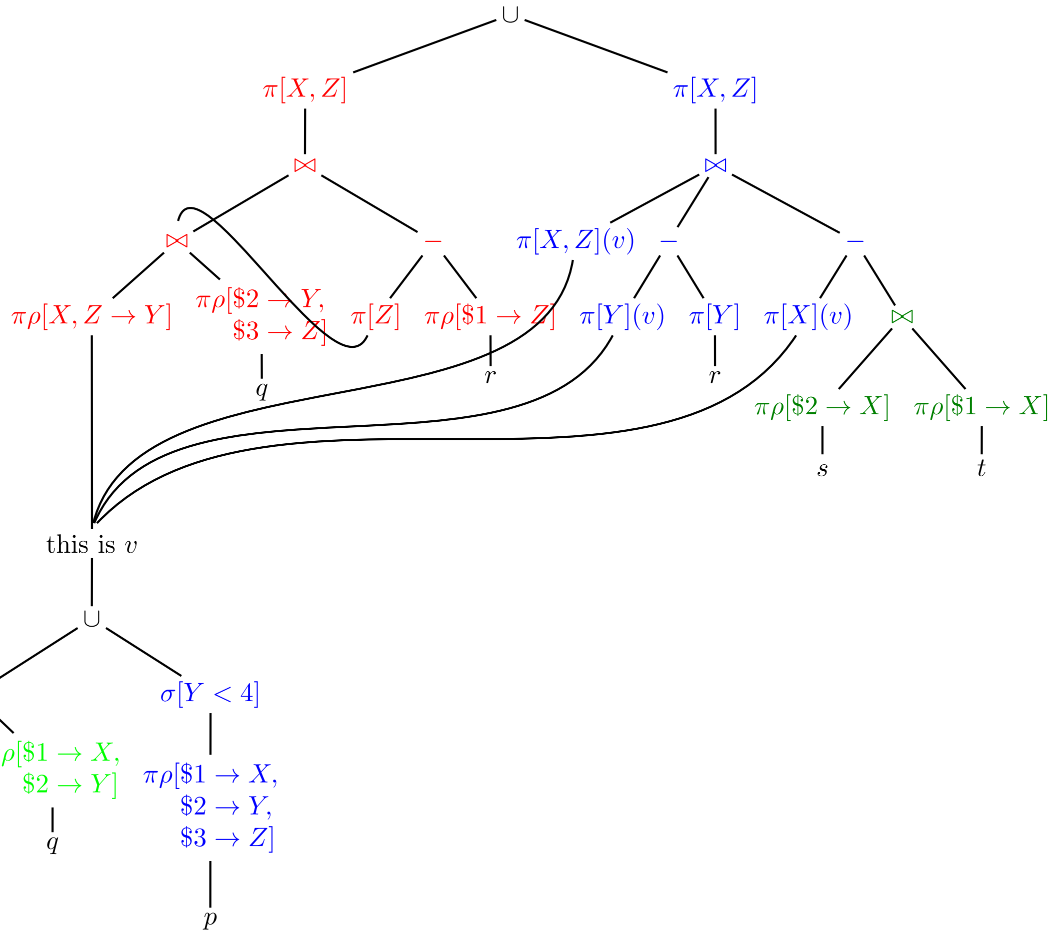
```
res(X,Z) :- v(X,_,_Y), q(_,_Y,Z), ¬r(Z,_).
res(X,Z) :- v(X,_Y,Z), ¬r(_Y,_), ¬w(X).
v(X,Y,Z) :- p(Z,_,X), q(X,Y,_).
v(X,Y,Z) :- p(X,Y,Z), Y<4.
w(X) :- s(_,X), t(X,_).
```

  where p/3, q/3, r/2, s/2, t/2 are EDB relations, v/3, w/1 are IDB relations (views).
  Give the algebra expression that corresponds to the res predicate.

- For each $C_i(X_1, \ldots, X_{k_i})$ and $D_i(X_1, \ldots, X_{k_i})$, there is an equivalent algebra expression $E_i = \rho[\ldots](\pi[\ldots](R_i))$ (note that R_i may be a complex expression if R_i is an EDB predicate) with format $(X_1, \ldots, X_{m_i})$ that selects the relevant attributes/variables/columns and renames them to $X_1, \ldots, X_{k_i}$.
  Safety implies that all variables that occur in any of the D_i also occur in at least one of the C_i.
  Let $C := E_1 \bowtie \ldots \bowtie E_m$. Then,

$$\pi[\mathsf{Vars}(B)](C \bowtie (\pi[\mathsf{Vars}(D_{m+1})](C) - D_{m+1}) \bowtie \ldots \bowtie (\pi[\mathsf{Vars}(D_{m+n})](C) - D_{m+n})) \qquad (*)$$

  is the required expression.
  (Note that this is analogous to the RANF to Algebra transformation; cf. Proof "Calculus to Algebra" in the lecture.)
- – comparison atoms of the form X_1 *op* X_j or X_i *op* c: selections applied to $(*)$.
  – If two rules define the same predicate symbol (i.e., have the same head): union.
- Subtrees/intermediate results can sometimes be used twice. This is supported by algebraic optimization and tabling.



Exercise 3 (Stratified Datalog)

Give an example for the nonmonotonicity of the stratified semantics,

show that for a stratifiable program P there can be multiple minimal models.

Consider the program P as follows:

```
borders(Y,X,Z) :- borders(X,Y,Z).    % make it symmetric.
reachable(X,Y) :- borders(X,Y,_).
reachable(X,Y) :- reachable(X,Z), borders(Z,Y,_).
unreachable(X,Y) :- country(X), country(Y), not reachable(X,Y).
```

The reachable predicate contains e.g. the pairs (D,F), (D,SGP), ..., The unreachable contains the pairs (D,USA), (D,BR), ..., (M,CY) (Malta, Cyprus, both are countries located on islands).

- If e.g. the fact borders(P,USA) is added to the database, –amongst others– (D,USA) is contained in reachable, and thus no more in unreachable.
- Cf. Lecture: the stratified model $\mathcal{M}$ of Mondial+P is a minimal model, i.e., there is no other model that is a proper subset of $\mathcal{M}$. On the other hand,

$\mathcal{M}'' = \mathcal{M} \cup \{\text{reachable(M,CY)}\} \setminus \{\text{unreachable(M,CY)}\}$, ist also a model, and it is also minimal (again, there is no proper subset of it that is also a model).

$\mathcal{M}''$ ist not "nice" because it contains "unfounded" reachable tuple.

In the rst of the lecture, some more notions of models will be considered: $\mathcal{M}''$ is neither well-founded (i.e., each atom in it is has some derivation), nor stable (i.e., it does not reproduce itself).

Exercise 4 (Datalog-Anfragen an Mondial: Schweizer Sprachen) Give Datalog programs for the following queries against the Mondial database. Compare with the same queries in the algebra and in the relational calculus.

a) All codes of countries in which some language is spoken that is also spoken in Switzerland.
b) All codes of countries in which only languages are spoken that are not spoken in Switzerland.
c) All codes of countries in which only languages are spoken that are also spoken in Switzerland.
d) All codes of countries in which all languages are spoken that are spoken in Switzerland.

```
:- auto_table.
:- include(mondial).

aufgA(C) :- language(C,_X,_), language('CH',_X,_).
aufgB(C) :- country(_,C,_,_,_,_), not aufgA(C).
nonCHLgCtry(C) :- language(C,_L,_), not language('CH',_L,_).
onlyCHLgCtry(C) :- country(_,C,_,_,_,_), not nonCHLgCtry(C).
chLgMissing(C) :- country(_,C,_,_,_,_), language('CH',_L,_), not language(C,_L,_).
noCHLgMissing(C) :- country(_,C,_,_,_,_), not chLgMissing(C).
?- aufgA(C).
?- aufgB(C).
?- onlyCHLgCtry(C).   % note: also countries with no language!
?- noCHLgMissing(C).
```

Exercise 5 (Datalog-Anfragen an Mondial: Landlocked)

- Give a Datalog program that returns the names of all countries that have no coast.
- Give a Datalog program that returns the names of all countries that have no coast and that have no neighbor country that has any coast.
- Give the dependency graph of your program.

```
:- auto_table.
:- include(mondial).

borders(Y,X,L) :- borders(X,Y,L).
coast(C) :- geo_sea(S,C,P).
landlocked(C) :- country(_,C,_,_,_,_), not coast(C).
hasnonlandlockedneighbor(C) :- landlocked(C), borders(C,C2,_), not landlocked(C2).
landlandlocked(C) :- landlocked(C), not hasnonlandlockedneighbor(C).
```

Asking `?- hasnonlandlockedneighbor(C)` yields many countries several times, e.g., MK (Macedonia) three times since `C2` can be bound by three ways to coastal neighbors: AL, GR, BG.

This can be avoided by a Prolog cut in the "subquery" that searches for possible `C2` bindings:

```
:- auto_table.
:- include(mondial).

borders(Y,X,L) :- borders(X,Y,L).
coast(C) :- geo_sea(S,C,P).
landlocked(C) :- country(_,C,_,_,_,_), not coast(C).
%hasnonlandlockedneighbor(C) :- landlocked(C), borders(C,C2,_), not landlocked(C2).
hasnlln(C) :- landlocked(C), hasnlln2(C).
hasnlln2(C) :- borders(C,C2,_), not landlocked(C2),!.
landlandlocked(C) :- landlocked(C), not hasnlln(C).
```

Exercise 6 (Aggregation in Datalog/XSB) Define the aggregation operators in XSB in a module `aggs.P`.

The syntax of the comparison predicates and of the arithmetic operators is given in Sections 3.10.5 (Inline Predicates) and 4.3 (Operators) of the XSB Manual Part I.

Then use `aggs.P` for answering the following queries in Datalog:

a) Give for each country the name and the number of neighbors.
b) Give the name of the country that has the highest number of neighbors (and how many).
c) Give the average area of all continents (to test avg).
d) Give the average latitude and longitude of all cities.

```
:- table avg/2.

sum(X,[H|T]) :- sum(Y,T), H \= null, Y \= null, X is H + Y.
sum(H,[H|T]) :- sum(null,T), H \= null.
sum(X,[null|T]) :- sum(X,T).
sum(null,[]).

?- sum(N, [1,2,3]).

count(X,[H|T]) :- count(Y,T), H \= null, X is Y + 1.
count(X,[null|T]) :- count(X,T).
count(0,[]).

?- count(N, [1,2,3]).

avg(X,L) :- sum(Y,L), count(C,L), Y \= null, C \= 0, X is Y / C.
avg(null,L) :- sum(Y,L), Y = null.
avg(null,L) :- count(C,L), C = 0.
avg(null,[]).

?- avg(N, [1,2,3]).

min(Y,[H|T]) :- min(Y,T), H \= null, Y \= null, H > Y.
min(H,[H|T]) :- min(Y,T), H \= null, Y \= null, H =< Y.
min(H,[H|T]) :- min(null,T), H \= null.
min(X,[null|T]) :- min(X,T).
min(null,[]).
```

```
max(Y,[H|T]) :- max(Y,T), H \= null, Y \= null, H =< Y.
max(H,[H|T]) :- max(Y,T), H \= null, Y \= null, H > Y.
max(H,[H|T]) :- max(null,T), H \= null.
max(X,[null|T]) :- max(X,T).
max(null,[]).
```

```
:- auto_table.
:- table neighbourscount/2, neighbourscount2/2, maxneighbourscount/1.
:- include(mondial).
:- include(aggs).

borders(X,Y) :- borders(X,Y,_).
borders(X,Y) :- borders(Y,X,_).

neighbours(X,NList) :- bagof(Y,borders(X,Y),NList).
neighbourscount(C,N) :- neighbours(C,NList), count(N,NList).

?- neighbourscount(_C,N), country(CName,_C,_,_,_,_).
?- neighbourscount(C,N), N > 10.
% oder kurz auch so:
neighbourscount2(C,N) :- bagof(Y,borders(C,Y),NList), count(N,NList).

%neighbourscounts(CList) :- bagof(N,C^neighbourscount(C,N),CList).
maxneighbourscount(M) :- bagof(_N,_C^neighbourscount(_C,_N),_CList), max(M,_CList).

% ?- maxneighbourscount(N).
% 16
?- maxneighbourscount(N), neighbourscount(_C,N), country(CName,_C,_,_,_,_).
?- neighbourscount(_C,N), country(CName,_C,_,_,_,_), maxneighbourscount(N).

% ?- avg(N,[9562488,45095292,8503474,30254708,39872000]).
% N = 26657592.4000
% ?- bagof(_Area,_CN^continent(_CN,_Area),_AreaList), avg(AvgArea,_AreaList).
% AvgArea = 26657592.4000

% ?- city('Stuttgart',_,_,_,Long,Lat).
% ?- bagof(Long,A^B^C^D^E^city(A,B,C,D,Long,E),_LongList), avg(AvgLong,_LongList).
% ?- bagof(Lat,A^B^C^D^E^city(A,B,C,D,E,Lat),_LatList), avg(AvgLat,_LatList).
avglonglat(AvgLong,AvgLat) :-
     bagof(_Long,_A^_B^_C^_D^_E^city(_A,_B,_C,_D,_Long,_E),_LongList), avg(AvgLong,_LongList),
     bagof(_Lat,_A^_B^_C^_D^_E^city(_A,_B,_C,_D,_E,_Lat),_LatList), avg(AvgLat,_LatList).
```
