Exercise 1, Sheet 2 :

Rel Algebra $\subseteq$ Datalog

→ Structural Induction over the expressions of the Relational Algebra

- R induction base
  $\boxed{\frac{A}{a}}$ " "minimal tables"
- $\sigma, \pi, \bowtie, (\varrho)$ "SPJR algebra" ≡ conjunctive queries
- $\cup, \setminus, \cap, \div$

disjunction negation

Jun 3-10:09

- R → need to know the arity R/n

$$res(X_1, \dots X_n) :- R(X_1, \dots, X_n).$$

- $\boxed{\frac{A}{a}}$

$$res(X) :- X = a .$$

- $\sigma[\alpha](p)$

  ↳ induction: there is an program p whose result relation is equiv. to p ≙ $p(X_1, \dots X_n)$

$$res(X_1 \dots X_n) :- p(X_1, \dots X_n), \alpha .$$

for conj. α
for disj → Union

Jun 3-10:24

- $\pi[X_{i_1}, \ldots X_{i_k}](p)$

  [by induction $p(X_1 \ldots X_n)$

$res(X_{i_1} \ldots X_{i_k}) \text{ :- } p(X_1, \ldots X_n)$

- join ..., no before: cartesian product

$$P \times Q$$

$p(X_1 \ldots X_n)$
$q(Y_1 \ldots Y_m)$

$res(\underbrace{X_1, \ldots X_n, Y_1 \ldots Y_m}_{= \{X_1..X_n\} \cup \{Y_1..Y_m\}}) \text{ :- } p(X_1 \ldots X_n),\ q(Y_1 \ldots Y_m)$

$\rightsquigarrow$ join: $P \bowtie Q \equiv \sigma[\text{common column names}]$ $X_{ij} = Y_{ks}$ when "X_{ij}" = "Y_{ks}"

$$\times(P, q)$$

- p: $p(Z_1..Z_k, X_{k+1}, \ldots X_n)$
  q: $q(Z_1 \ldots Z_k, Y_{k+1} \ldots Y_m)$

$$X_{k+i} \neq Y_{k+j} \quad \forall i,j > 0$$

$res(Z_1 \ldots Z_k, X_{k+1} \ldots X_n, Y_{k+1}, \ldots Y_m) \text{ :- }$
$p(Z_1..Z_k\ X_{k+1}..X_n),\ q(Z_1..Z_k, Y_{k+1}..Y_m)$

- renaming:

$\delta[X_1 \to Y_1, \ldots X_{i_k} \to Y_{i_k}](p)$

column names change

.... Datalog has no column names,

$res(X_1 \ldots X_n) = p(X_1 \ldots X_n)$

Jun 3-10:30

- disjunction/union

$$\cup(p, q)$$

same arity
p/n q/n

"$\cup$"

$$\sigma[\alpha \vee \beta](p) = \sigma[\alpha](p) \cup \sigma[\beta](p)$$

$res(X_1 \ldots X_n) \text{ :- } p(X_1, \ldots X_n).$
$res(X_1 \ldots X_n) \text{ :- } q(X_1, \ldots X_n).$

"$\alpha \vee \beta$"

$res(X_1 \ldots X_n) \text{ :- } p(X_1 \ldots X_n),\ \alpha.$
$res(X_1 \ldots X_n) \text{ :- } p(X_1 \ldots X_n),\ \beta.$

Jun 3-10:35

$$res(X_1 \ldots X_n) :- p(X_1 \ldots X_n), \text{not } q(X_1 \ldots X_n)$$

p, q same arity

rule is safe!

recall: more complex negation

p where not exists (sq.)

⇒ rewrite into "\" in algebra

⇒ in Datalog simpler:

$$res(X_1 .. X_n) :- p(X_1 \ldots X_n), \text{not } sq(\_, \_, X_i, \ldots)$$

Jun 3-10:46

Exercise 2: Datalog to Algebra

go the same way: structural induction over?

Over the structure of Datalog programs

- individual rules $h \leftarrow b$ — different forms of body?
- program as a whole: set of rules $r_1 \ldots r_n$

Simplest form(s) of Datalog rules

$$res(X_1 \ldots X_n) :- p(X_1 \ldots X_n).$$

Algebra equiv: p

$$res(X_1 \ldots \cancel{X_i} \ldots X_n) :- p(X_1, \ldots, \_, \ldots X_n).$$

$\uparrow X_i$

$$\pi[\$_1 \ldots \$_{i-1}, \$_{i+1} \ldots \$_n](p)$$

$$res(X_1, \ldots X_i \ldots X_n) :- p(X_1, \ldots, c, \ldots X_n).$$

↑ i-th column

$$\sigma[X_i = c](p)$$

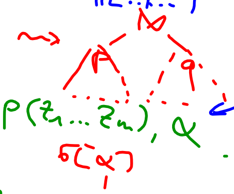
Jun 3-10:53

next complex case: two predicates in the body (or more → induction over $\mathbb{N}$)

- $res(Z_1, \dots Z_k, \cancel{X_1} \dots X_m, \cancel{Y_1} \dots Y_n) :-$

  $p(Z_1 \dots Z_k, X_1 \dots X_m), q(Z_1 \dots Z_k, Y_1 \dots Y_n).$

  $\pi[\dots]$



- $res(Z_1 \dots Z_n) :- p(Z_1 \dots Z_n), \alpha.$

  comparison, e.g. $Z_i = Z_j$, $Z_i = c$

  $\sigma[\alpha]$

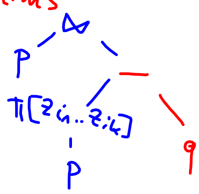
- negative literals

  we have only safe rules → all vars must be bound in some positive literal

  → embed it into the induction over the number of body literals

  $res(Z_1 \dots Z_n) :- p(Z_1 \dots Z_m), \neg q(Z_{i_1} \dots Z_{i_k}).$

  … relational minus



… now we are done with individual rules

Jun 3-11:00

Consider now programs = $\{r_1 \dots r_k\}$

induction over $\mathbb{N}$

1 rule ✓

2 rules:

$h_1 \leftarrow b_1$
$h_2 \leftarrow b_2$

- if $h_1 \neq h_2$ then there are two result views
- if $h_1 = h_2$ (= predicate name and arity are the same)

  $h_1(X_1 \dots X_n) :- b_1.$
  $h_2(X_1 \dots X_n) :- b_2.$

induction:

Jun 3-11:11

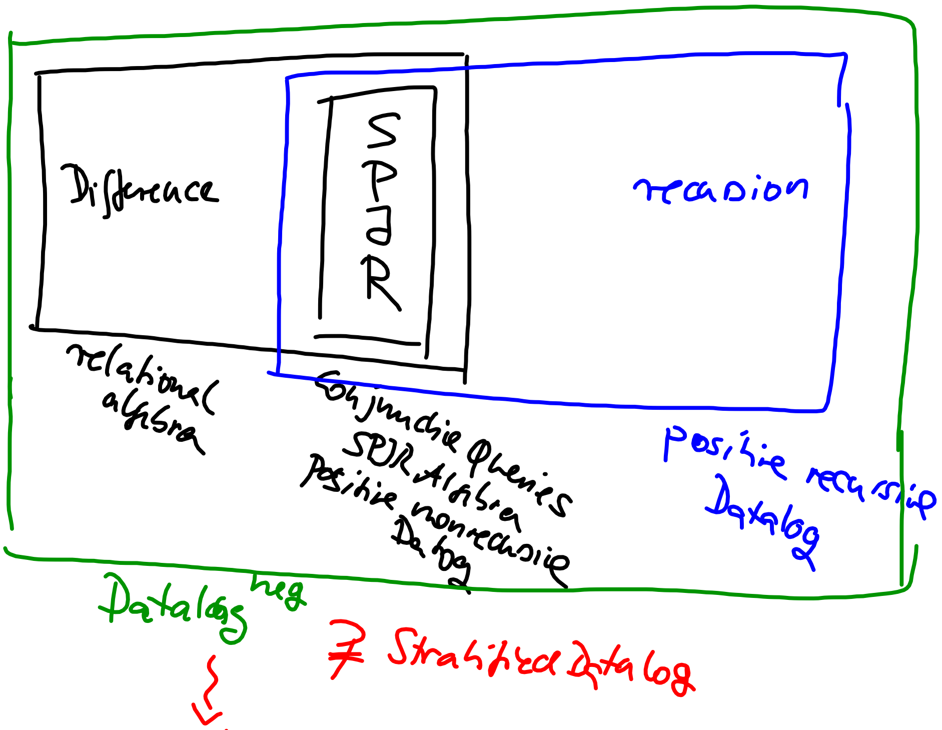
up to now: omitted one detail!

proof works only if head predicate $\neq$ any of the body predicates!

$\Rightarrow$ we can translate any NONRECURSIVE rule/program into the rel. Algebra

$\Rightarrow$ for recursive programs: not possible!

Jun 3-11:15



$\nexists$ Stratified Datalog

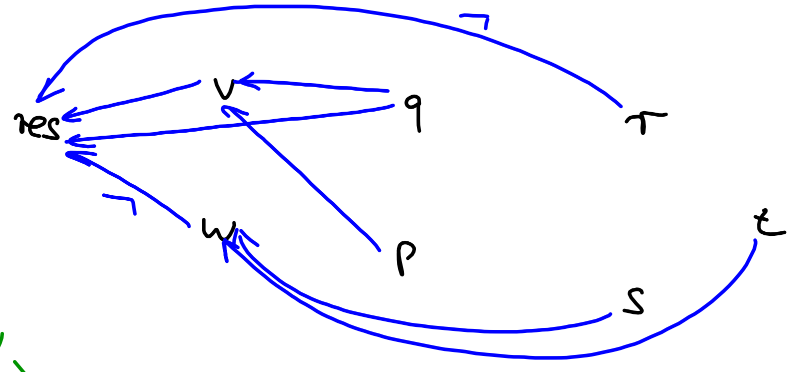
well-founded Datalog

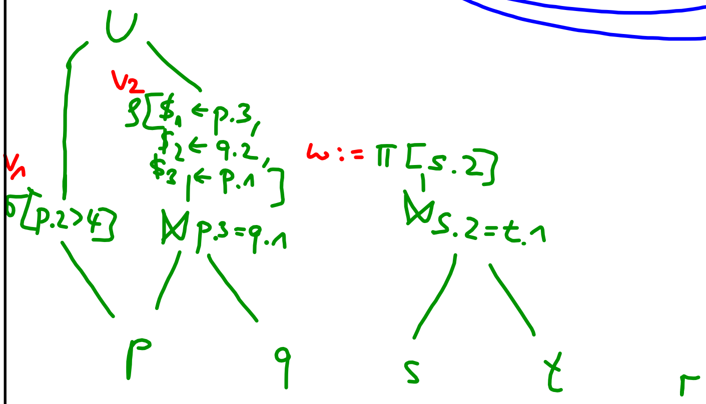
Stable semantics

Jun 3-11:21

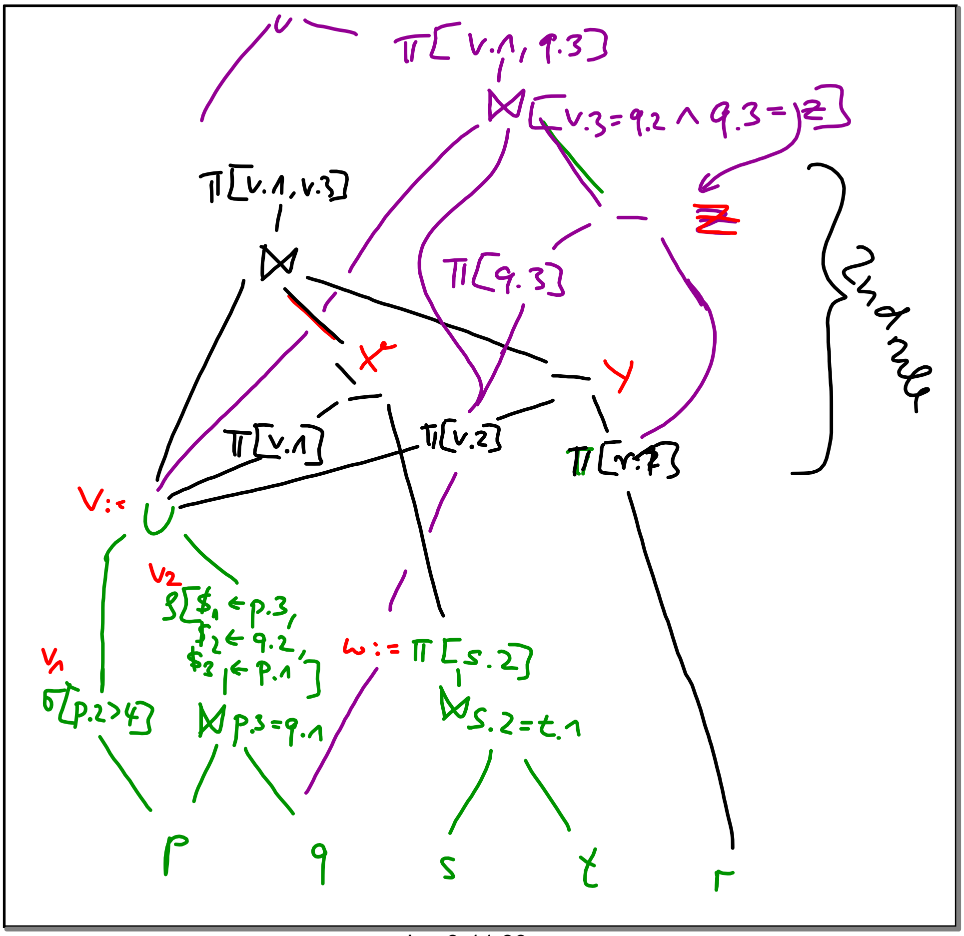
Second part of Ex 2:

Start with the dependency graph





Jun 3-11:29



Jun 3-11:39