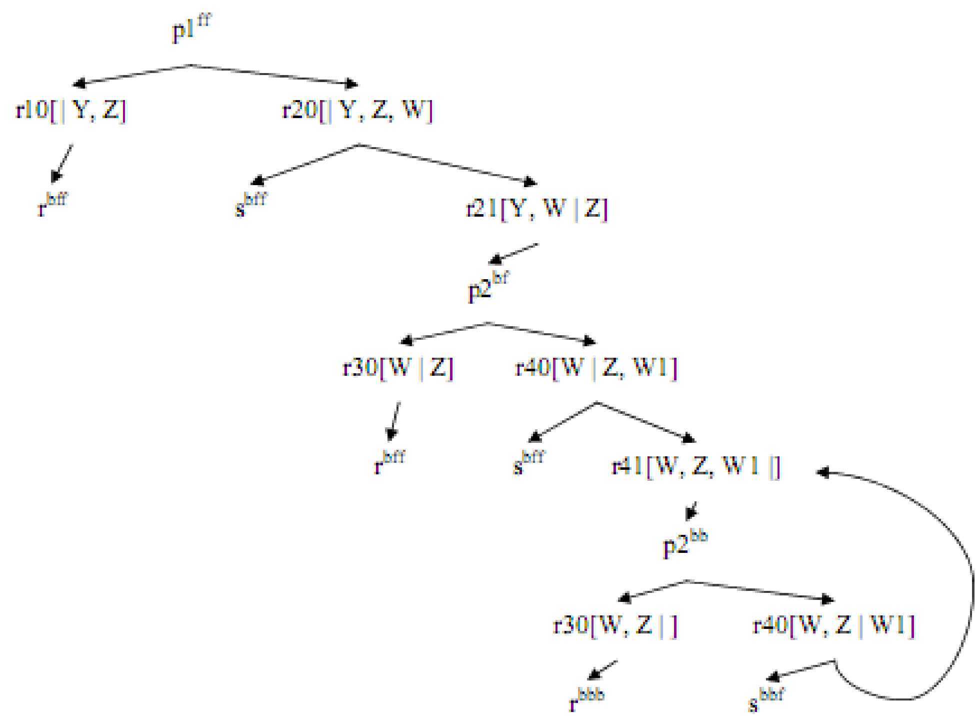
1. Daný je program

p(X, Y, Z) ← r(X, Y, Z).
p(X, Y, Z) ← s(X, Y, W), p(W, Z, Z).
s dotazom ? ← p(1, Y, Z).

a) Transformujte program jednoduchou mágiou. (Dôsledne rektifikujte.) (2 body)

Rektifikaciu začneme s dotazom (obsahuje konštantu):
? ← p(1, Y, Z). p1(Y, Z).
p1(Y, Z) ← r(1, Y, Z).
p1(Y, Z) ← s(1, Y, W), p(W, Z, Z) p2(W, Z).
p2(W, Z) ← r(W, Z, Z).
p2(W, Z) ← s(W, Z, W1), p2(W1, Z).
p(X, Y, Z) ← r(X, Y, Z).
p(X, Y, Z) ← s(X, Y, W), p(W, Z, Z).
Rektifikovaný program pre daný dotaz:
? ← p1(Y, Z).
r1: p1(Y, Z) ← r(1, Y, Z).
r2: p1(Y, Z) ← s(1, Y, W), p2(W, Z).
r3: p2(W, Z) ← r(W, Z, Z).
r4: p2(W, Z) ← s(W, Z, W1), p2(W1, Z).

RGG pre rektifikovaný program:



Jednoduchá mágia:

? ← p1(Y, Z).
m_p1.
m_p2bf(W) ← sup21(Y, W).
m_p2bb(W1, Z) ← sup41(W, Z, W1).
sup10 ← m_p1.

sup20 ← m_p1.
sup30bf(W) ← m_p2bf(W).
sup40bf(W) ← m_p2bf(W).
sup30bb(W, Z) ← m_p2bb(W, Z).
sup40bb(W, Z) ← m_p2bb(W, Z).
sup21(Y, W) ← sup20, s(1, Y, W).
sup41(W, Z, W1) ← sup40bf(W), s(W, Z, W1).
sup41(W, Z, W1) ← sup40bb(W, Z), s(W, Z, W1).
p1(Y, Z) ← sup10, r(1, Y, Z).
p1(Y, Z) ← sup21(Y, W), p2(W, Z).
p2(W, Z) ← sup30bf(W), r(W, Z, Z).
p2(W, Z) ← sup30bb(W, Z), r(W, Z, Z).
p2(W, Z) ← sup41(W, Z, W1), p2(W1, Z).

Jednoduchá mágia po zjednodušeni:

? ← p1(Y, Z).
m_p2bf(W) ← s(1, Y, W).
m_p2bb(W1, Z) ← m_p2bf(W), s(W, Z, W1).
m_p2bb(W1, Z) ← m_p2bb(W, Z), s(W, Z, W1).
p1(Y, Z) ← r(1, Y, Z).
p1(Y, Z) ← s(1, Y, W), p2(W, Z).
p2(W, Z) ← m_p2bf(W), r(W, Z, Z).
p2(W, Z) ← m_p2bb(W, Z), r(W, Z, Z).
p2(W, Z) ← m_p2bf(W), s(W, Z, W1), p2(W1, Z).
p2(W, Z) ← m_p2bb(W, Z), s(W, Z, W1), p2(W1, Z).

b) Transformujte program zovšeobecnenou mágiou. (2 body)

? ← p(1, Y, Z).
m_p(1, Y, Z).
m_p(W, Z, Z) ← sup21(X, Y, Z, W).
sup10(X, Y, Z) ← m_p(X, Y, Z).
sup20(X, Y, Z, W) ← m_p(X, Y, Z).
sup21(X, Y, Z, W) ← sup20(X, Y, Z, W), s(X, Y, W).
p(X, Y, Z) ← sup10(X, Y, Z), r(X, Y, Z).
p(X, Y, Z) ← sup21(X, Y, Z, W), p(W, Z, Z).

Zovšeobecnená mágia po zjednodušeni:

? ← p(1, Y, Z).
m_p(1, _, _).
m_p(W, Z, Z) ← m_p(X, Y, Z), s(X, Y, W).
p(X, Y, Z) ← m_p(X, Y, Z), r(X, Y, Z).
p(X, Y, Z) ← m_p(X, Y, Z), s(X, Y, W), p(W, Z, Z).

c) Porovnajte vypočet seminaivnou evaluáciou pre pôvodny program a pre niektorý magický program pre nasledujúcu EDB: {r(X, Y, Z) = {[5, 6, 6], [5, 6, 7], …, [5, 6, n]}, s(X, Y, Z) = {[1, 2, 3], [3, 4, 5]}}. (2 body)

Pôvodny program:

0.iterácia
p = {}
1.iterácia
p = {[5, 6, 6], [5, 6, 7], …, [5, 6, n], [3, 4, 6]}
2.iterácia
p = {[5, 6, 6], [5, 6, 7], …, [5, 6, n], [3, 4, 6]}

Program dosiahol pevný bod. Výsledkom dotazu je Ø.

Jednoduchá mágia:

0.iterácia
p1 = {}, p2 = {}, m_p2bf = {}, m_p2bb = {}
1.iterácia
p1 = {}, p2 = {}, m_p2bf = {3}, m_p2bb = {[5, 4]}
2.iterácia
p1 = {}, p2 = {}, m_p2bf = {3}, m_p2bb = {[5, 4]}

Program dosiahol pevný bod. Výsledkom dotazu je Ø.

Zovšeobecnená mágia:

0.iterácia
m_p = {}, p = {}
1.iterácia
m_p = {[1, _, _]}, p = {}
2.iterácia
m_p = {[1, _, _], [3, Z, Z]}, p = {}
3.iterácia
m_p = {[1, _, _], [3, Z, Z]}, p = {}

Program dosiahol pevný bod. Výsledkom dotazu je Ø.

Príklad 2

Daný je program:

p(X) ← a(X),¬q(X).
q(X) ← b(X,Y), p(Y).
q(X) ← c(X).

a EDB = {a={<1>,<3>}, b={<1,2>, <2,3>}, c={<3>}}.

Nájdite:

1. Všetky modely. (2b)

Inštanciované vyčistené pravidlá:

~~p(0) ← a(0),¬q(0).~~
p(1) ← ~~a(1),~~¬q(1).
~~p(2) ← a(2),¬q(2).~~
p(3) ← ~~a(3),~~¬q(3).
~~q(1) ← c(1).~~
~~q(2) ← c(2).~~
q(3) ← ~~c(3).~~
~~q(0) ← b(0,0), p(0).~~ /* Ďalej už len rozumne inštanciované pravidlá... */
q(1) ← ~~b(1,2),~~ p(2).
q(2) ← ~~b(2,3),~~ p(3).

Kvôli prehľadnosti ešte raz:

p(1) ← ¬q(1).
p(3) ← ¬q(3).
q(3).
q(1) ← p(2).
q(2) ← p(3).

Všetky „byrokraciou dovolené“ modely: EDB +

p(1)	T	F	T	T	F	F	T	T	F	*T*	*T*	T	T	F	F
p(2)	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**
p(3)	**T**	**T**	**F**	**F**	**F**	**F**	**T**	**T**	**T**	**F**	**F**	**F**	**F**	**F**	**F**
q(1)	*T*	*T*	*T*	*T*	*T*	*T*	T	F	T	**F**	**F**	**T**	**T**	**T**	**T**
q(2)	*T*	*T*	T	F	T	F	*T*	*T*	*T*	T	F	T	F	T	F
q(3)	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*T*

Matematicky EDB plus konjunkcia uvedených troch implikácii.
Princíp byrokracie zakazuje konštanty (inštancie) nevyskytujúce sa v databáze alebo v pravidlách.

2. Stabílne modely. (2b)
   EDB ∪ {p(1), q(3)}

3. Well-founded model. (2b)

EDB+

p(1)	F	T	T	**T**
p(2)	F	F	F	**F**
p(3)	F	T	F	**F**
q(1)	F	F	F	**F**
q(2)	F	F	F	**F**
q(3)	F	T	T	**T**

Poznámky:

1. Largest unfounded set = {p(2), p(3), q(1), q(2)}.
2. Lokálne stratifikovaný: stratum0={ p(2), q(1), q(3)}, stratum1={p(1),p(3),q(2)}.
3. Ak well-founded model je dvojhodnotový, potom je to aj minimálny stabilný model.
4. Ak program je lokálne stratifikovaný, potom existuje jediný stabílny model, ktorý je aj well-founded model.
5. Matematika dovoľuje aj modely, ktoré obsahujú aj konštanty nevyskytujúce sa v EDB. p(X), q(X). Pre taketo X môže platiť čokoľvek [,p(X), q(X), p(X)∧q(x)]. Princíp byrokracie nám zakazaje tieto modely.

Príklad 3 (Levy – Sagivov test)

Q_1: $p(Y) \leftarrow a(X, Y), a(Y, Z), \neg a(Z, X).$
Q_2: $p(X) \leftarrow a(X, Y), \neg a(Y, X).$

Najprv nech oba dotazy majú rovnaké hlavy. Dosadím U za X a X za Y.

Q_1: $p(X) \leftarrow a(U, X), a(X, Z), \neg a(Z, U).$
Q_2: $p(X) \leftarrow a(X, Y), \neg a(Y, X).$

Kanonické databázy sú všetky databázy nad doménou, ktorá má toľko prvkov, koľko pohlcovaný dotaz obsahuje premenných.

Pre $Q_1 \supseteq Q_2$ vystačíme s dvomi hodnotami, t.j všetky orientované grafy (so slučkami) s dvomi uzlami. Vyberiem graf s jedinou hranou a(0,1). Q_2 dá odpoveď {X=0}, Q_1 vráti {} prázdnu množinu. Teda pohltenie neplatí.

Pre $Q_2 \supseteq Q_1$ na dokázanie platnosti pohltenia potrbujeme otestovať všetky orientované grafy (so slučkami) s tromi uzlami. Vyberieme graf s hranami a(0,1), a(1,2), a(1,0), a(2,1). Q_1 vráti {X=1,X=2} a Q_2 vráti prázdnu množinu {}. Ani toto pohltenie neplatí.

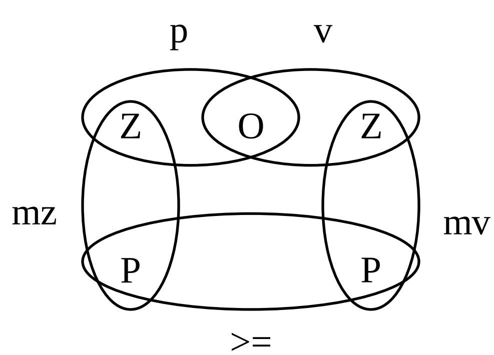
Príklad 4

Daná je báza dát s tromi binárnymi reláciami
mzdy(Zamestnanec, Plat),
pracuje(Zamestnanec, Oddelenie),
vedie(Zamestnanec, Oddelenie).
Predpokladáme, že v relácii *vedie* platí funkčná zavislosť *Oddelenie* → *Zamestnanec*, vo zvyšných dvoch reláciách *Zamestnanec* je kľúč.

1. Sformulujte SQL dotaz na zamestnancov (Z, P), ktori zarábajú aspoň o 30% viac ako vedúci oddelenia v ktorom pracujú. **(1b)**

```
select p.Zamestnanec, mz.Plat
from pracuje p, vedie v, mzdy mz, mzdy mv
where p.Oddelenie = v.Oddelenie and p.Zamestanec = mz.Zamestnanec and
v.Zamestnanec = mv.Zamestnanec and mz.Plat >= 1.3 * mv.Plat
```

2. Nakreslite hypergraf dotazu. **(2b)**



3. Optimalizujte. **(2b)**

Hypergraf dotazu je cyklický, takže treba použiť Wong-Youssefiho algoritmus. Odtrhneme hranu v, lebo v je zrejme menšia relácia než p:
$p := \Pi_{p.Zamestnanec} (p \ltimes v)$
$mv := \Pi_{Plat} (mv \ltimes v)$
$res(G) = \Pi_{p.Zamestnanec,\ p.Plat} (v \bowtie res\ (G - v))$

Odtrhneme hranu p od mz:
$p := \Pi_{p.Zamestnanec} (p \ltimes v)$
$mv := \Pi_{Plat} (mv \ltimes v)$
$mz := mz \ltimes p$

$res(G) = \Pi_{p.Zamestnanec,\ p.Plat} (\Pi_{p.Zamestnanec,\ p.Plat,\ v.Plat} (v \bowtie p) \bowtie res\ (G - v - p))$

Teraz musíme odtrhnúť selekčnú hranu:
$p := \Pi_{p.Zamestnanec} (p \ltimes v)$
$mv := \Pi_{Plat} (mv \ltimes v)$
$mz := mz \ltimes p$

$res(G) = \Pi_{p.Zamestnanec,\ p.Plat} (\Pi_{p.Zamestnanec,\ p.Plat,\ v.Plat} (v \bowtie p) \bowtie$
$\Pi_{mz.Zamestnanec,\ mz.Plat} (\sigma_{mz.Plat >= 1.3 * mv.Plat} (res\ (G - v - p - <=))))$

Zvyšok hypergrafu tvoria dve komponenty súvislosti:
$p := \Pi_{p.Zamestnanec} (p \ltimes v)$
$mv := \Pi_{Plat} (mv \ltimes v)$
$mz := mz \ltimes p$
$res(G) = \Pi_{p.Zamestnanec,\ p.Plat} (\Pi_{p.Zamestnanec,\ p.Plat,\ v.Plat} (v \bowtie p) \bowtie$
$\Pi_{mz.Zamestnanec,\ mz.Plat} (\sigma_{mz.Plat >= 1.3 * mv.Plat} (mz \times \Pi_{mv.Plat} (mv))))$

Vo výslednom dotaze je rozumné nahradiť kartézsky súčin a následnú selekciu joinom:
$\mathbf{res(G) = \Pi_{p.Zamestnanec,\ p.Plat} (\Pi_{p.Zamestnanec,\ p.Plat,\ v.Plat} (v \bowtie p) \bowtie}$
$\mathbf{\Pi_{mz.Zamestnanec,\ mz.Plat} (mz \bowtie_{mz.Plat >= 1.3 * mv.Plat} \Pi_{mv.Plat} (mv)))}$

Príklad 5

Máte realizovať minimálny (čo do počtu serverov) distribuovaný systém, ktorý umožňuje čítať pri výpadku troch serverov a zapisovať pri výpadku dvoch serverov bez čakania na obnovu spadnutých serverov.

1. Koľko serverov treba použiť? Zdôvodnite. (2b)
2. Popíšte distribúciu a replikáciu dát, stanovte minimálne quora na čítanie a písanie. (2b)
3. Aké dôsledky na funkčnosť transakcií bude mať pri Vašom návrhu súčasný výpadok troch a viac serverov? (2b)

Podmienky pre fungovanie:

$$\begin{array}{rl} -n + q_r + q_w & \geq 1 \\ n - q_r & \geq 3 \\ \underline{n - q_w} & \underline{\geq 2} \\ n & \geq 6 \end{array}$$

Podmienku $2q_w - n \geq 1$ zatial ignorujem.

Potrebujeme 6 serverov.

Pri minimálnom počte serverov data musia byť replikované na všettkých serveroch. Quorum pre čítanie je 3. Quorum pre zápis je 4. Podmienka $2q_w - n \geq 1$ je splnená.
Pri výpadku troch serverov transakcie, ktoré zapisovali musia čakať, alebo byť abortované. Abort je možno lepší, lebo píšucé transakcie môžu držať množstvo zamknutých dát. “Read only” transakcie bežia ďalej. Pri výpadku 4 a viac serverov system sa zastaví. Ak veríme, že sa výpadkom nič zlé nestalo, môžu všetky transakcie čakať. Abort je bezpečný v každom prípade.