Príklad 1 (Ullmann). Daný je datalógový program

$p(X) \leftarrow p(f(X)).$

$p(X) \leftarrow a(X).$

Predpokladáme, že a je nasledovná EDB relácia: a(0). a(f(f(2))). a(f(f(f(3)))).

1. Ako sa tento program chová pri seminaivnom výpočte cieľa p^f (bez používania pravidla subsumcie) ? (2b)

Pri seminaivnom výpočte beží výpočet takto (diferencia je zvýraznená tučným písmom):

p^0 = {**0, f(f(2)), f(f(f(3)))**}.

p^0 = {0, f(f(2)), f(f(f(3))), **f(2), f(f(3))**}.

p^1 = {0, f(f(2)), f(f(f(3))), f(2), f(f(3)), **2, f(3)**}.

p^2 = {0, f(f(2)), f(f(f(3))), f(2), f(f(3)), 2, f(3), **3**}.

p^3 = {0, f(f(2)), f(f(f(3))), f(2), f(f(3)), 2, f(3), 3}.

Tu výpočet skončí.

2. Ako sa chová QRGT algoritmus pre výpočet cieľa p^f ? (2b)

Fronta a výstup QRGT (Queue-based Rule Goal Tree) algoritmu:

$queue^0$: p(X).

$queue^1$: p(f(X)), a(X).

Output: 0, f(f(2)) f(f(f(3))).

$queue^2$: p(f(f(X))), a(f(X)), a(X).

Output: 0, f(f(2)), f(f(f(3))), f(2) f(f(3)).

$queue^3$: p(f(f(f(X)))), a(f(f(X))), a(f(X)), a(X).

Output: 0, f(f(2)), f(f(f(3))), f(2), f(f(3)), 2, f(3).

$queue^4$: p(f(f(f(f(X))))), a(f(f(f(X)))), a(f(f(X))), a(f(X)), a(X).

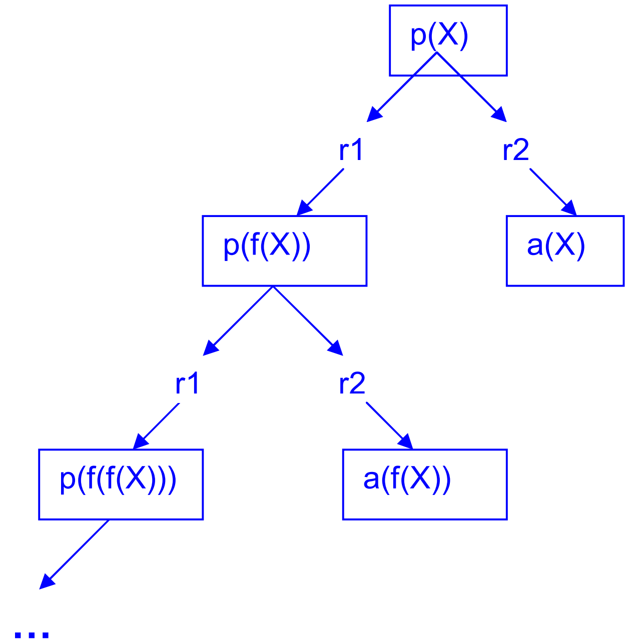
Output: 0, f(f(2)), f(f(f(3))), f(2), f(f(3)), 2, f(3), 3.

$queue^5$: p(f(f(f(f(f(X)))))), a(f(f(f(f(X))))), a(f(f(f(X)))), a(f(f(X))), a(f(X)), a(X).

Output: 0, f(f(2)), f(f(f(3))), f(2), f(f(3)), 2, f(3), 3.

...

QRGT algoritmus sa zacyklí, bude stále expandovať podvýraz X na f(X). Strom (RGT) sa rozvíja do nekonečna.



Po niekoľkých krokoch bude a(f(f(...(X)...)) vracať prázdnu množinu. Program expandGoal-expandRule sa napriek tomu zacyklí.

3. Urobte zovšeobecnenú magickú transformáciu programu pre výpočet cieľa p^f. (3b)

Z rule-Goal Graph z predošlej úlohy:

1. skupina
m_p(f(X)) ← $sup_{1,0}$(X).
m_a(X) ← $sup_{2,0}$(X).

2. skupina
$sup_{1,0}$(X) ← m_p(X).
$sup_{2,0}$(X) ← m_p(X).

3. skupina

4. skupina
p(X) ← $sup_{1,0}$(f(X)), p(f(X)).
p(X) ← $sup_{2,0}$(X), a(X).

5. skupina
m_p(X).

Pravidlá 1. a 2. skupiny môžeme vynechať. Potom v pravidlách 4. skupiny treba substituovať m_p(f(X)) za $sup_{1,0}$(X) a m_p(X) za $sup_{2,0}$(X). Výsledný program:
m_p(f(X)) ← m_p(X).
p(X) ← m_p(f(X)), p(f(X)).
p(X) ← m_p(X), a(X).
m_p(X).

4. Riešte otázky (1) a (2) pre transformovaný program. (2b)

Pri seminaivnom výpočte beží výpočet tak ako bez magickej transformácie.

Tu je niekoľko krokov QRGT výpočtu magického programu, bez subsumcie:
$queue^0$: p(X).
$queue^1$: p(f(X)), m_p(X), a(X).
Output: 0, f(f(2)) f(f(f(3)).
$queue^2$: p(f(f(X))), m_p(f(X)), m_p(f(f(X))), m_p(X), a(f(X)), a(X).
Output: 0, f(f(2)), f(f(f(3))), f(2) f(f(3)).
$queue^3$: p(f(f(f(X)))), m_p(f(f(f(X)))), m_p(f(f(X))), m_p(f(X)), m_p(X), a(f(f(X))), a(f(X)), a(X).
Output: 0, f(f(2)), f(f(f(3))), f(2) f(f(3)), 2, f(3).
...

Oba programy sa zacyklia v dôsledku nekonečného výpočtu predikátu m_p.

5. V čom pomáha subsumcia v úlohe 4? (1b)

(Pravidlo subsumcie: Pri výpočte nepridávame n-ticu, ktorá je špeciálnym prípadom n-tice, ktorá sa už v relácii vyskytuje.)

Podľa pravidla subsumcie f(X) $\subseteq$ X, a prvé pravidlo, spôsobujúce zacyklenie (t.j. pravidlo m_p(f(X)) $\leftarrow$ m_p(X)), možno vynechať.

Príklad 2. Daný je program

p(X) $\leftarrow \neg$ q(X).
q(X) $\leftarrow \neg$ p(X).
p(1).
q(0).

Kde p a q sú predikáty nad množinou {0, 1, 2}.

1. Nájdite všetky stabilné modely uvedeného programu. (3b)

Skúmajme inštanciovaný program (prvé dve pravidlá) nad univerzom {c}, kde c je konštanta:

p(c) $\leftarrow \neg$ q(c).
q(c) $\leftarrow \neg$ p(c).

Toto je konjunkcia dvoch implikácií (ekvivalentná p(c)$\vee$ q(c)). Z pravdivosti p(c) nevieme odvodiť nič o pravdivosti q(c). Rovnako z pravdivosti q(c) nevieme odvodiť nič o pravdivosti p(c). Avšak z nepravdivosti p(c) vyplýva pravdivosť q(c), a z nepravdivosti q(c) vyplýva pravdivosť p(c).

Z týchto úvah je vidieť, že pre každú konštantu c musí každý stabilný model nad množinou obsahujúcou c obsahovať buď p(c) alebo q(c) (alebo oboje). Ďalej, každý (stabilný) model obsahuje p(1) a q(0) (to hovoria posledné dve pravidlá programu). Takže všetky stabilné modely nad množinou {0, 1, 2} sú:

{q(0), p(1), q(2)},
{q(0), p(1), p(2)},
{q(0), p(1), p(2), q(2)},
{q(0), p(0), p(1), q(2)},
{q(0), p(0), p(1), p(2)},
{q(0), p(0), p(1), p(2), q(2)},
{q(0), p(1), q(1), q(2)},
{q(0), p(1), q(1), p(2)},
{q(0), p(1), q(1), p(2), q(2)},
{q(0), p(0), p(1), q(1), q(2)},
{q(0), p(0), p(1), q(1), p(2)},
{q(0), p(0), p(1), q(1), p(2), q(2)}.

2. Nájdite „well-founded" model. (2b)

Well-founded model sa dá nájsť napríklad tabuľkovou metódou:

p(0)	F	F	F
p(1)	T	T	T
p(2)	F	T	F
q(0)	T	T	T
q(1)	F	F	F
q(2)	F	T	F

Už sa to opakuje. Takže vo well-founded modeli platí p(1) a q(0), neplatí p(0) a q(1), neznáme (unknown) sú q(2) a p(2).

Príklad 3. Dané sú konjunktívne dotazy:

Q_1: p(X, Y) $\leftarrow$ p(X, Z), p(Z, Y).
Q_2: p(X, Y) $\leftarrow$ p(X, A), p(B, Y).
Q_3: p(X, Y) $\leftarrow$ p(X, A), p(B, Y), q(A, B).

1. Zistite vzťahy pohltenia:
   Q_1 ? Q_2 (1b)
   Q_1 ? Q_3 (1b)
   Q_2 ? Q_3 (1b)

Platí $Q_1 \subseteq Q_2$, vďaka homomorfizmu $A \rightarrow Z$, $B \rightarrow Z$ atď. Opačné pohltenie neplatí.
Neplatí $Q_1 \subseteq Q_3$, ani $Q_3 \subseteq Q_1$.
Platí $Q_3 \subseteq Q_2$, ale neplatí $Q_2 \subseteq Q_3$.

Ako sa zmenia vzťahy pohltenia za predpokladu, že q je relácia rovnosti? Vysvetlite! (2b)

V tom prípade sa dá Q_3 písať ako p(X, Y) $\leftarrow$ p(X, Z), p(Z, Y).
Platí $Q_1 \subseteq Q_3$, aj $Q_3 \subseteq Q_1$.
Platí $Q_3 \subseteq Q_2$, ale neplatí $Q_2 \subseteq Q_3$.

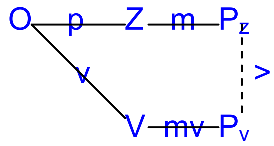
Alternatívne riešenie:
Rektifikujeme Q_1: p(X,Y) $\leftarrow$ p(X,Z_1), p(Z_2), Z_1=Z_2.
Platí $Q_1 \subseteq Q_3$, aj $Q_3 \subseteq Q_1$.
Platí $Q_3 \subseteq Q_2$, ale neplatí $Q_2 \subseteq Q_3$.

Priklad 4. Daná je báza dát s tromi binárnymi reláciami
mzdy(Zamestnanec, Plat), pracuje(Zamestnanec, Oddelenie), vedie(Zamestnanec, Oddelenie).
Predpokladáme, že v relácii vedie platia funkčné závislosti O → Z, Z → O.

1. Sformulujte SQL dotaz na zamestnancov (Z, P), ktorí zarábajú viac ako ich vedúci. (1b)

select M.zamestnanec, M.plat **from** mzdy M, mzdy MV, vedie V, pracuje P
where V.oddelenie = p.oddelenie **and** V.zamestnanec = MV.zamestnanec **and** P.zamestnanec = m.zamestnanec **and** M.plat > MV.plat

2. Nakreslite hypergraf dotazu ! (1b)



3. Zostrojte strom dotazu ! (1b)

Naivná konštrukcia stromu zodpovedá algebraickému výrazu:

$$\prod_{Z,Pz} \sigma_{Pz>Pv} ((\prod_{V\leftarrow Z} v) \bowtie p \bowtie (\prod_{V\leftarrow Z,\ Pv\leftarrow P} m) \bowtie (\prod_{Pz\leftarrow P} m)).$$

Vnútorné projekcie sú technicky potrebné operácie, aby joiny boli prirodzené spojenia.

4. Optimalizujte ! (3b)

Hypergraf je cyklický. Vieme použiť (okrem originálnych nápadov) Wong – Yussefiho algoritmus alebo optimalizáciu stromu výrazu.
Relačné hrany p a v pretinajú iné relačné hrany a nepretínajú žiadnu selekčnú hranu sú teda kandidáti na odstránenie. Ani jedna z nich nie je malá (v zmysle definície). Poradie ich odstránenia môžeme vybrať ľubovolne. Napr.: v , p teraz už musíme vybrať selekčnú hranu, čím sa hypergraf rozpadne na dve nezávislé komponenty. Výsledok ako algebraický výraz vyzerá nasledovne $\prod_{Z,Pz} (v \bowtie (p \bowtie (\sigma_{Pz>Pv}\, m \times mv)$. Vzhľadom na platné závislosti semijoiny nič nerdukujú (budeme ich ignorovať).
Výsledný výraz vyzerá ako anti optimalizácia. Quel však počíta použitím multijoinov. Výsledný optimalizovaný výpočet je:

```
for each O,V from v do
          { find Pv = P from m where m.Z = V;
            for each Z from p where p.O=O do
                    { find Pz = P from m where m.Z = Z;
                      if Pz > Pv then output(Z, Pz);
                    }
          }
```

Uvedná schéma je dobrá, ak Z je primárny kľúč v relácií m a Z je sekundárny index v relácii p. Tieto predpoklady sú nielen splniteľné, ale vysoko pravdepodobné pri dobrom návrhu databázy.

Druhá možnosť je určiť poradie joinov (na základe intuitívneho odhadu veľkosti relácií a predpokladaných funkčných závislostí) a pretláčať selekcie a projekcie k listom na strome výrazu.

Príklad 5. (Christianov protokol). Klient mal na svojich hodinách 14:10:56 (hodiny:minúty:sekundy) v okamihu, keď posielal serveru žiadosť o zaslanie presného času (predpokladá sa, že server má na svojich hodinách presný čas a že odpovedá okamžite). V okamihu, keď mal klient na svojich hodinách 14:11:00, dostal od servera odpoveď „presný čas je 14:10:48“. Popíšte čo najpresnejšie, čo má klient urobiť, aby počas 20 nasledujúcich sekúnd bez ďalšej komunikácie nastavil svoje hodiny čo najpresnejšie. (5b)

Trvalo 4 sekundy, kým sa správa dostala od klientovi k serveru a nazad. Rozumný predpoklad je, že prenos správy od klientovi k serveru trvá rovnako dlho ako od servera ku klientovi, t.j. 2 sekundy (tento čas sa merať nedá). To znamená, že údaj 14:10:48 prišiel ku klientovi s oneskorením 2 sekundy.
T.j. okamihu, keď klient mal na svojich hodinách čas 14:11:00, bol presný čas 14:10:50. Klient nesmie posunúť svoje hodiny o 10 sekúnd dozadu. Aby sa klient synchronizoval so serverom, potrebuje v tejto chvíli svoje hodiny spomaliť počas nasledujúcich 20 (reálnych) sekúnd.