Príklad 1 (7b)

Daný je program: p(X, Y) ← r(X, Y).
p(X, Y) ← p(X, Z), ¬p(Z, Y).

Kde EDB je r = {<2,1>, <3,2>} nad doménou d = {1,2,3,c}.

1. Nájdite jeho minimálne stabílne modely ! **(3b)**

Na inštanciované pravidlá hneď aplikujeme GLT. Pritom do modelu dáme najprv EDB a postupne model rozširujeme. Tu je fragment inštanciovaného programu:

r(2, 1).
r(3, 2).
p(2, 1) ← ~~r(2, 1)~~.
p(3, 2) ← ~~r(3, 2)~~.
p(2, 2) ← ~~p(2, 1), ¬p(1, 2)~~. /* Ďalšie pravidlá pre p(2, 2) už netreba. */
p(2, 3) ← ~~p(2, 1), ¬p(1, 3)~~. /* Ďalšie pravidlá pre p(2, 3) už netreba. */
p(2, c) ← ~~p(2, 1), ¬p(1, c)~~. /* Ďalšie pravidlá pre p(2, c) už netreba. */
~~p(3, 1) ← p(3, 2), ¬p(2, 1).~~ /* Lebo p(2, 1) platí. */
~~p(3, 2) ← p(3, 2), ¬p(2, 2).~~ /* Lebo p(2, 2) platí. */
~~p(3, 3) ← p(3, 2), ¬p(2, 3).~~ /* Lebo p(2, 3) platí. */
~~p(3, c) ← p(3, 2), ¬p(2, c).~~ /* Lebo p(2, c) platí. */

Je očividné, že r(2, 1), r(3, 2), p(2,1), p(2,2), p(2,3), p(2,c) a p(3,2) je stabilný model. Je totiž invariantný vzhľadom na GLT, a aplikáciou programu nedokážeme odvodiť už nič iné. Napríklad nedokážeme odvodiť p(1, Y), Y∈{1, 2, 3, c}, lebo jediné pravidlo ktoré by nám to dovolilo, je p(1, Y) ← p(1, Z), ¬p(Z, Y). Takže p(1, Y) (hlavu) vieme odvodiť len ak pridáme do modelu niektoré p(1, Z). Takto rozšírený model však určite nebude *minimálny* stabilný model. Analogicky je možné argumentovať, že nedokážeme odvodiť p(c, Y), Y∈{1, 2, 3, c}.

Ostáva ukázať, že r(2, 1), r(3, 2), p(2,1), p(2,2), p(2,3), p(2,c) a p(3,2) je *minimálny* stabilný model. p(2, 1) a p(3, 2) musia byť v každom modeli, lebo priamo vyplývajú z EDB. Ak však je v stabilnom modeli p(2, 1) (a to platí v každom modeli, lebo to priamo vyplýva z EDB) a nie je v ňom napríklad p(1, 2), tak potom v ňom je p(2, 2). Toto je vidieť z GLT nad čiastočne inštanciovaným programom hore. Analogické tvrdenie platí pre p(2, 3) a p(2, c).

2. Nájdite „well-founded“ model ! **(3b)**

Použijeme alternujúci pevný bod:

	0	1	2	3
p(1, 1)	F	F	F	F
p(1, 2)	F	F	F	F
p(1, 3)	F	F	F	F
p(1, c)	F	F	F	F
p(2, 1)	**T**	T	T	T
p(2, 2)	F	**T**	T	T
p(2, 3)	F	**T**	T	T
p(2, c)	F	**T**	T	T
p(3, 1)	F	**T**	**F**	F
p(3, 2)	**T**	T	T	F
p(3, 3)	F	**T**	**F**	F
p(3, c)	F	**T**	**F**	F
p(c, 1)	F	F	F	F
p(c, 2)	F	F	F	F
p(c, 3)	F	F	F	F
p(c, c)	F	F	F	F

Teda platí r(2, 1), r(3, 2), p(2,1), p(2,2), p(2,3), p(2,c) a p(3,2), ostatné hodnoty predikátu p sú FALSE. Je to dvohodnotový well-founded model a teda je aj stabilným modelom.

3. Nájdite všetky modely ! **(1b)**

Každý model zodpovedá inštanciácii premenných U, V, X, Y, Z $\in$ {1, 2, 3, c}, pre ktorú je splnená formula:

$\forall$U, V, X, Y, Z$\in${1, 2, 3, c}

(r(2, 1) $\wedge$ r(3, 2) $\wedge$ (p(U, V) $\vee$ $\neg$ r(U, V)) $\wedge$ ($\neg$ p(X, Z) $\vee$ p(Z, Y) $\vee$ p(X, Y)))

$\equiv$

$\forall$X, Y, Z$\in${1, 2, 3, c}

(r(2, 1) $\wedge$ r(3, 2) $\wedge$ p(2, 1) $\wedge$ p(3, 2) $\wedge$ ($\neg$ p(X, Z) $\vee$ p(Z, Y) $\vee$ p(X, Y)))

Ľudskou rečou povedané: musí platiť r(2, 1), r(3, 2), p(2,1) a p(3,2) a pre každé pridané p(X, Z) musíme pridať buď p(Z, Y) alebo p(X, Y).

Modely, kde pridávame do EDB sú matematicky možné, ale v databázach s nimi neuvažujeme. (Princíp byrokracie.)

Príklad 2 (6b)

Daný je logický program

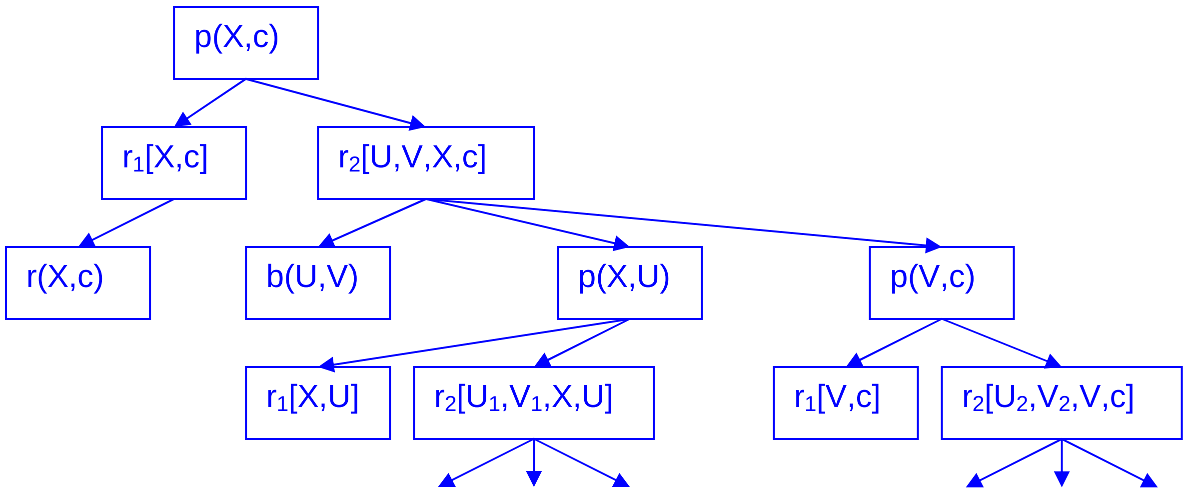
r_1: p(X, Y) ← r(X, Y)

r_2: p(X, Y) ← b(U,V), p(X,U), p(V, Y) .

?- p(X, c)

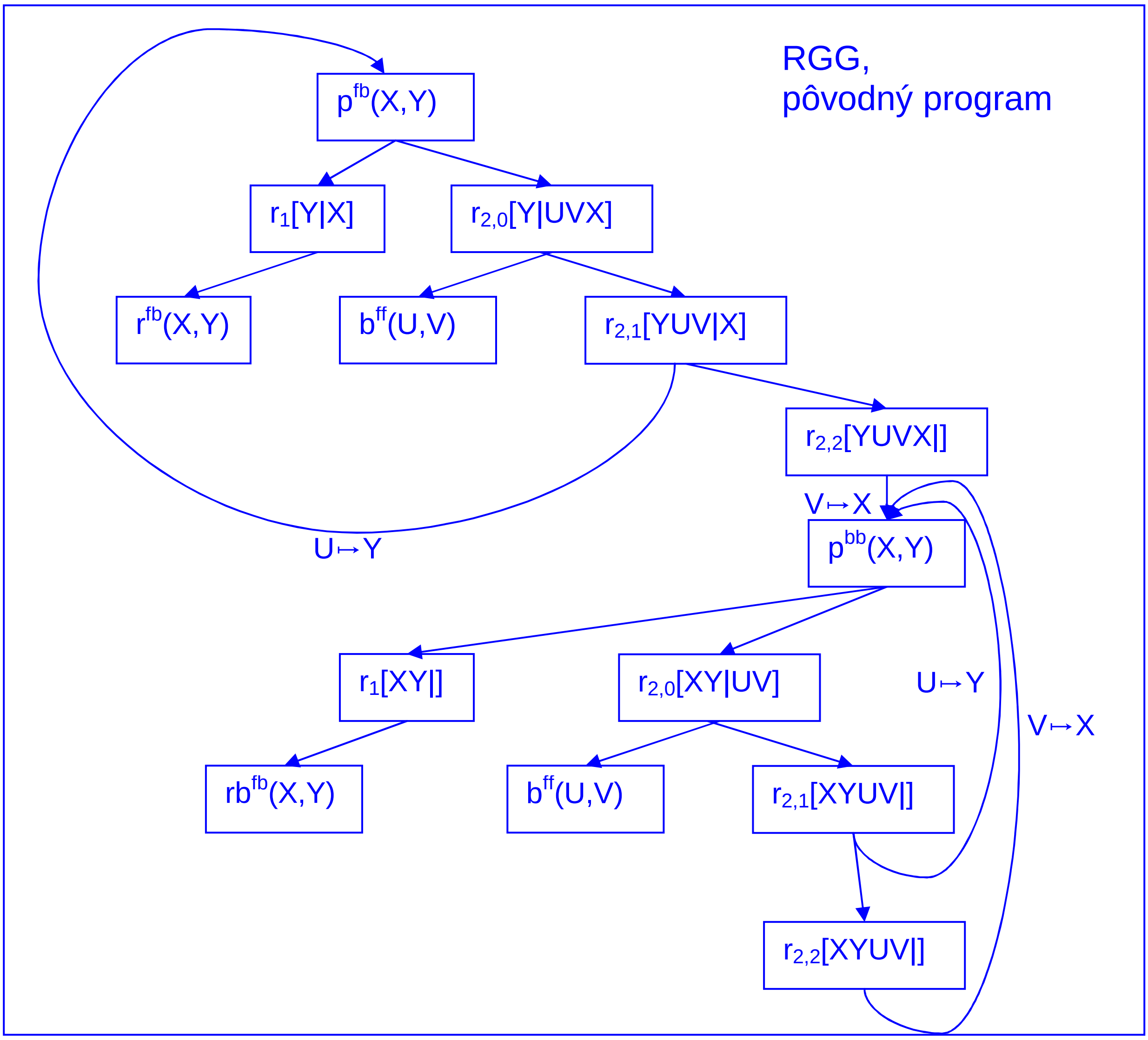
kde r a b sú extenzionálne predikáty .

1. Nakreslite strom cieľov a pravidiel ! **(1b)**



Premenné v pravidlových cieľoch vznikajú podľa nasledujúceho pravidla: Do pravidiel substituujeme ešte sa nevyskytujúce premenné a existujúci cieľ v strome „unifikujeme" s takto vzniklým pravidlom.

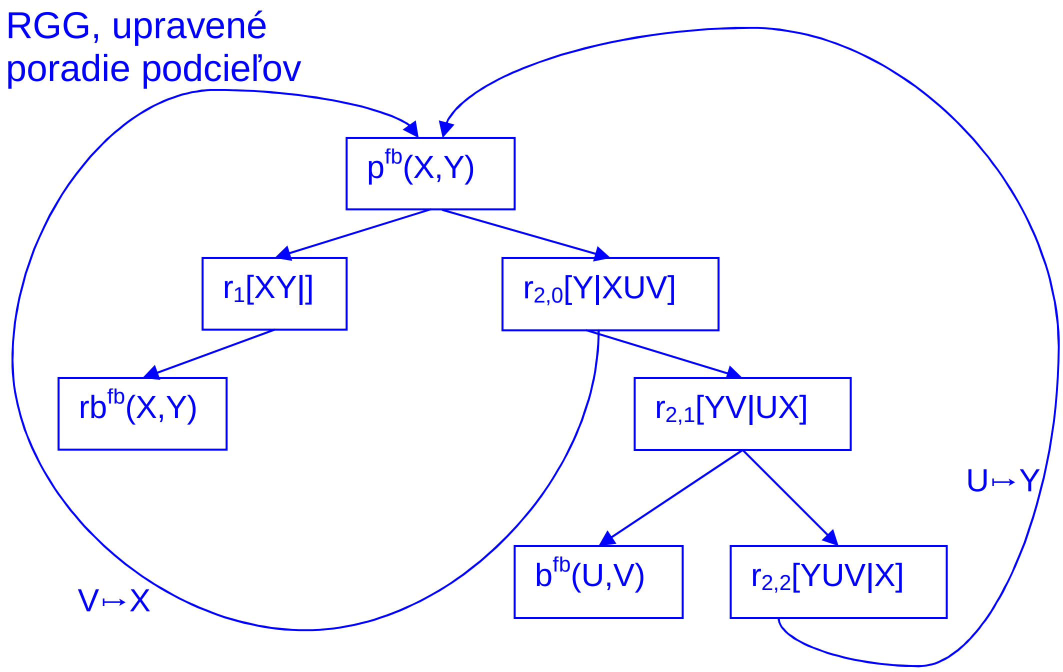
2. Nakreslite graf cieľov a pravidiel ! **(1b)**



Premenovanie premenných na niektorých hranách je pomôcka pre lepšie porozumenie. V knihe a prednáškach to nebolo.

3. Preusporiadajte ciele pre zjednoznačnenie väzieb (ozdôb)! **(2b)**

Druhé pravidlo: $p^{fb}(X,Y) \leftarrow p^{fb}(V, Y),\ b^{fb}(U,V), p^{fb}(X,U)$.



4. Urobte zovšeobecnenú magickú transfomáciu ! **(2b)**

m_p(c). semeno (V.skupina).

m_p(U) $\leftarrow$ **m_p(Y), p(V,Y), b(U,V)**. (I. skupina)

Rozpisané:

$sup_{2,1}(V,Y) \leftarrow m\_p(Y), p(V,Y)$.

$sup_{2,2}(U,Y) \leftarrow sup_{2,1}(V,Y), b(U,V)$. /* Premennú V v ďalšom výpočte nepotrebujeme. */

p(X, Y) $\leftarrow$ **m_p(Y), r(X, Y).** (IV. skupina)

/* Nulté pomocné predikáty sú o ničom, možno ich rovno nahradiť magickým predikátom. Ani rozpis $sup_{1,1}(X, Y) \leftarrow m\_p(Y), r(X, Y)$. a $p(X,Y) \leftarrow sup_{1,1}(X, Y)$ nepotrebujeme. */

p(X,Y) $\leftarrow$ **m_p(Y), p(V,Y), b(U,V), p(X,U).** (IV. skupina)

Rozpísané:

$sup_{2,1}(V,Y) \leftarrow m\_p(Y), p(V,Y)$.

$sup_{2,2}(U,Y) \leftarrow sup_{2,1}(V,Y), b(U,V)$.

$p(X,Y) \leftarrow sup_{2,2}(U,Y), p(X,U)$.

Vidíme, že druhej a tretej skupiny pravidiel sa možno často zbaviť. (Vlastne vždy, ale nie vždy je to výhodné pre výpočet.)

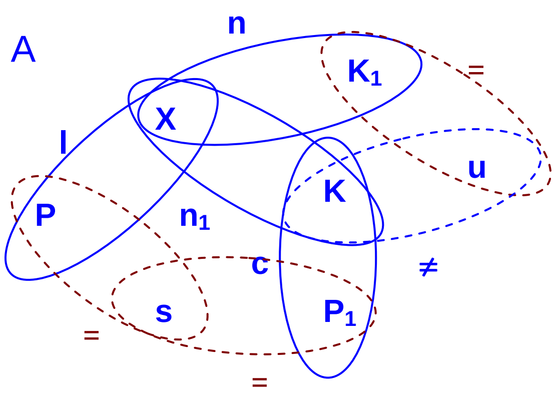
Príklad 3 (6 b)

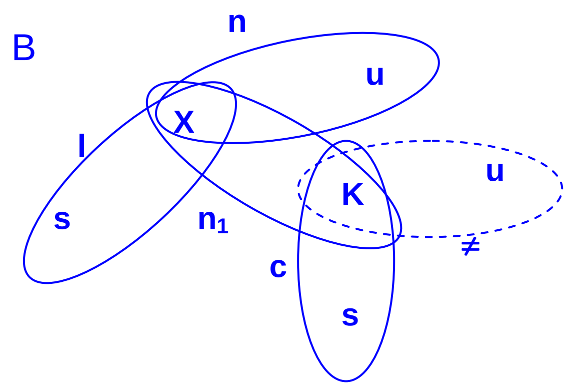
Daná je relačná databázová schéma:
Čapujú(Krčma, Pivo)
Ľúbi(Pijan, Pivo)
Navštívil(Pijan, Krčma).
Sformulujte SQL dotaz na všetkých pijanov, ktorí ľubia pivo značky Stein, navštívili krčmu „U Steina“ a aspoň jednu inú krčmu, kde čapujú pivo značky Stein.

slelect l.Pijan from Ľúbi l where l.Pivo = "stein" intersect
select n.Pijan from Navštívil n where n.Krčma = "u steina" intersect
select n.Pijan from Navštívil n, Čapujú c
where n.Krčma = c.Krčma and c.Krčma <> "u steina" and c.Pivo = "stein";

1. Znázornite hypergraf dotazu, preložte do algebry. **(2b)**

Označenie: X Pijan, P Pivo, K Krcma, s stein, u u steina.





$$\prod_X \sigma_{P=s} l \cap \prod_X \sigma_{K=u} n \cap \prod_X (\sigma_{K\neq u \wedge P=s} c \bowtie n)$$

2. Zistite, či existuje úplný reduktor pre príslušný spojovací výraz. **(1b)**

Na obrázku A je podľa prednášky nakreslený hypergraf dotazu. Na obrázku B je ten istý hypergraf po odstránení selekcií na rovnosť. Pri odstraňovaní selekcií na rovnosť je užitočné ignorovať fakt, že konštanty v relačných hranách sú rovnaké a nevytvárať zbytočné cykly. Obrázok B je acyklický a teda má úplný reduktor. Všetky relačné uši v obrázku B sú malé.

3. Optimalizujte Wong Youssefiho algoritmom ! **(3b)**

Je prakticky jedno, na ktorom obrázku budeme vykonávať algoritmus. Ak sa rozhodneme pre obrázok B, nesmieme zabudnúť pred odstránením „malej“ relačnej hrany zaradiť príslušnú selekciu. Rozhodol som sa pre poradie: lcn_1. Pretože, ide o binárne relácie s jedným konštatným argumentom, ktorý nejde do výsledku, semijoiny aj joiny (po projekcii na zaujímave argumenty) sú prieniky. Postupne vypočítame ($\prod_X \sigma_{P=s} l \cap \prod_X \sigma_{K=u} n$) a $\prod_K (\sigma_{K\neq u \wedge P=s} c)$ a na záver overíme, či každý vybraný pijan navštivil niektorú z vybraných krčiem (stačí jednu).

Príklad 4 (6b)

a) Daná je tabuľka R(A, B, C).
Minimalizujte výraz: $\pi_{AB}(\sigma_{B=5}(R)) \bowtie \pi_{BC}(\pi_{AB}(R) \bowtie \pi_{AC}(\sigma_{B=5}(R)))$. **(3b)**

Tabló:

R	A	B	C
	A	B	
	A	5	
	A	B	
	A	5	C

Po minimalizácii zostane prvý riadok. Teda $\pi_{AB}(\sigma_{B=5}(R))$.

b) Daná je EDB pozostávajúca s binárnej relácie a(X,Y) a nekonečná množina konjunktívnych dotazov: Q_i: p(X) ← a(X,Z_1), a(Z_1,Z_2), …, a(Z_{i-1},Z_i), a(Z_i,X). Zistite pohltenie medzi dvojicami dotazov Q_i a Q_j. Nájdite logický program ekvivalentný zjednoteniu $\cup(0 \leq i < \infty)Q_i$. **(3b)**

Ekvivalentný logický program sa dá nájsť ľahko, ak si uvedomíme, čo tie dotazy Q_i počítajú. Prvky relácie a možno chápať ako hrany orientovaného grafu. Dotaz Q_i počíta, či v tom grafe existuje cyklus (cesta, ktorá končí v tom vrchole, v ktorom začala) dĺžky i. Pozor, tento cyklus nemusí byť nutne vrcholovo disjunktný (t.j. niektorý vrchol sa v tej ceste môže vyskytnúť viackrát.)
To nekonečné zjednotenie je dotaz, ktorý počíta, či ten graf obsahuje nejaký cyklus (t.j. cyklus ľubovoľnej dĺžky):

c(X, Y) ← a(X, Y).
c(X, Y) ← a(X, Z), c(Z, Y). /* Počíta cestu z X do Y. */
p(X) ← c(X, X).

Ďalej je zaujímavé (to nebolo v úlohe) sa pýtať na vzájomné pohltenia medzi dvojicami dotazov Q_i a Q_j:
Porovnáme Q_0: p(X) ← a(X, X). a

Q_1: p(X) ← a(X, Z_1), a(Z_1,X).

Zrejme $Q_0 \subseteq Q_1$, stačí Z_1 namapovať na X.
Všeobecne $Q_i \subseteq Q_j$, keď $j = k \times i$, pre $k \in \{0,1,2, \ldots, \}$. Opačné mapovania nefungujú, kvôli hlave. Názorná predstava: a(X, Y) je hrana a Q_n cyklus dĺžky n. Pohltenie dostaneme tým, že ten istý cyklus prejdeme niekoľkokrát.
Výpočet p(X) toho predošlého programu pri i+1. iterácii pohltí Q_i. To znamená, že náš program pohltí zjednotenie dotazov Q_i. (Platí aj opačné tvrdenie. Teda ten program po i+1 iteráciách vypočíta $\cup(0 \leq k \leq i)Q_k$.)

Príklad 5 (6b)

Uvažujte distribuovanú databázu bez replikovaných dát a predpokladajte, že jednotlivé uzly nepadajú. Predpokladajte, že na synchronizáciu transakcií sa používaju zámky, pričom sa rozlišuje medzi read-lock a write-lock.

a) Popíšte protokoly pre centralizovanú a distribuovanú správu zámkov pre horeuvedený systém. **(2b)**

V centralizovanom systéme prideľuje všetky zámky jeden centrálny uzol.
V distribuovanom systéme prideľuje zámky na dáta ten uzol, na ktorom sú tie dáta uložené.
V oboch prípadoch transakcia žiada o zámok svoj „domáci uzol", ktorý tú žiadosť preposiela uzlu, ktorý smie zámok prideliť (komunikácia medzi site-managers).
V oboch prípadoch sa používa dvojfázové zamykanie.

b) Predpokladajte, že horeuvedený systém používa centralizovanú správu zámkov. Možu transakcie skončiť uviaznutím? Svoju odpoveď ÁNO resp. NIE zdôvodnite, t.j. buď vysvetlite prečo uviaznutie nemôže nastať, alebo uveďte príklad uviaznutia. **(2b)**

Áno, uviaznutie môže nastať. Centrálny správca zámkov môže prideliť wl1(X), wl2(Y). Ak neskôr dostane požiadavky wl1(Y), wl2(X), tak obe transakcie uviaznu, lebo čakajú jedna na druhú.

Na druhej strane, existujú stratégie „schedulera", ktoré zabraňujú vzniku deadlocku. Napr. je možné udržovať graf čakania bez cyklov, alebo použiť niektorý z algoritmov wait or die či wait or wound. Takisto dvojfázový protokol s určeným poradím zamykania dát alebo s povinnosťou žiadať všetky zámky naraz stačia na vylúčenie uviaznutia.

c) Predpokladajte, ze horeuvedený systém používa distribuovanú správu zámkov, pričom každy z uzlov sa lokálne riadi striktným dvojfázovým protokolom. Môžu transakcie skončiť uviaznutím, ktoré žiaden z uzlov nedokáže lokálne detekovat? Svoju odpoveď ÁNO resp. NIE zdôvodnite, t.j. buď vysvetlite prečo takéto uviaznutie nemôže nastať, alebo uveďte príklad takéhoto uviaznutia. **(2b)**

Áno, takéto uviaznutie môže nastať. Nech SX je uzol, ktorý spravuje objekt X a nech SY je uzol rôzny od SX, ktorý spravuje objekt Y. Nech SX pridelí transakcii T1 wl1(X) a nech SY pridelí transakcii T2 wl2(Y). Teraz transakcia T1 žiada wl1(Y) u SY, a transakcia T2 žiada wl2(X) u SX. Žiaden z uzlov SX a SY nevie lokálne detekovať uviaznutie transakcií T1 a T2.

Vo všeobecnosti sa graf čakania rozdelí lokálne na niekoľko častí. Každá časť je acyklická, ale globálny graf obsahuje cyklus. Je jedno, či ten graf konštrujú servery vykonávajúce transakcie alebo servery spravujúce dáta, alebo oba typy serverov (obyčajne to robia servery spravujúce dáta). Pokiaľ nejaký koordinátor nevytvorí globálny graf čakania, nikto neodhalí uviaznutie.

Zdalo by sa, že vyššie uvedené stratégie wait or die alebo wait or wound problém riešia. Omyl! Na to, aby tieto protokoly fungovali, je treba globálny synchrónny čas vo všetkých uzloch. Lenže centrálny časový server alebo centrálny správca zámkov, to je prakticky jedno. Oboje je rovnako zlé.