

# DJ2 – vyjadřovací síla slajdy k přednášce NDBI001

**J. Pokorný**

**MFF UK**

# Sémantika DRK (1)

Předpoklady: výrazy dotazu  $\{x_1, \dots, x_k \mid A(x_1, \dots, x_k )\}$ ,  $A$  je formule DRK,

databáze  $\mathbf{R}^*$ ,  $dom$  je doména pro  $\mathbf{R}$ , *aktuální doména* formule  $A$ ,  $adom(A)$ , je množina hodnot z relací v  $A$  a konstant v  $A$ .

Tři problémy:

- ❖ potenciální možnost nekonečné odpovědi (v případě nekonečné  $dom$ )
- ❖ situace, kdy TRUE-ohodnocení volných proměnných není z databáze.
- ❖ jak implementovat vyhodnocení kvantifikace (v případě nekonečné  $dom$ ) v konečném čase.

# Sémantika DRK (2)

Sémantiky pro DRK, které problémy řeší.

- (i) neomezená interpretace s omezeným výstupem
- (ii) omezená interpretace
- (iii) doménově nezávislé dotazy

Značení: výsledek vyhodnocení dotazu  $D$  v neomezené interpretaci jako  $D^{\text{dom}}[\mathbf{R}^*]$ .

Pak:

- ❖ pro (i) je výsledek definován jako  $D^{\text{dom}}[\mathbf{R}^*] \cap \text{adom}^k$ , kde  $k$  je řád výsledné relace.
- ❖ pro (ii) pouze přes hodnoty z  $\text{adom}$ , tj.  $D^{\text{adom}}[\mathbf{R}^*]$ .

# Sémantika DRK (3)

D.:  $\{x \mid \neg R(A:x)\}$

⇒ Odpověď je závislá na  $dom(A)$ .

⇒ Výraz dotazu definuje pro různé domény různé dotazy.

Pz.: Doménově závislým může být i dotaz, který vrací  $\emptyset$ , např.

D.:  $\{x \mid \forall y R(x,y)\}$

v případě, že se kvantifikuje přes nekonečnou doménu.

Df.: Výraz dotazu nazveme **doménově nezávislým (definitním, určitým)**, jestliže odpověď na něj nezávisí na  $dom$ .

Dotazovací jazyk je **doménově nezávislý**, jestliže každý jeho výraz je doménově nezávislý. Výsledek dotazu  $D$  je roven  $D^{\text{dom}}[\mathbf{R}^*] = D^{\text{adom}}[\mathbf{R}^*]$ .

# Sémantika DRK (4)

⇒ vyhodnocení doménově nezávislého výrazu v neomezené interpretaci dává stejný výsledek jako v omezené interpretaci.

Př.:

¬ KNIHA(TITUL:'Úvod do DBS',AUTOR:a)  
NENÍ definitní.

$\exists ič (EXEMPLÁŘ(ič,d) \wedge VÝPŮJČKA(ič,č,z))$   
JE definitní

$\exists ič (EXEMPLÁŘ(ič,d) \vee VÝPŮJČKA(ič,č,z))$   
NENÍ definitní, jsou-li proměnné netypované nebo s příliš  
"širokými" typy

Tv.: (Di Paola 1969)

Definitnost A není rozhodnutelná.

⇒ jazyk doménově nezávislých formulí není rozhodnutelný.

Pz.: Relační algebra je doménově nezávislý jazyk.  
Dotazovací jazyky

# Sémantika DRK (5)

Značení DRK:

- ❖ v neomezené interpretaci s omezeným výstupem  $\text{DRK}^{\text{rout}}$ ,
- ❖ v omezené interpretaci  $\text{DRK}^{\text{lim}}$
- ❖ doménově nezávislé výrazy  $\text{DRK}^{\text{ind}}$ .

Tv.:  $\text{DRK}^{\text{rout}} \cong \text{DRK}^{\text{lim}} \cong \text{DRK}^{\text{ind}}$ . Navíc,

- (i) je-li  $D$  výraz DRK, pak existuje doménově nezávislý výraz  $D'$ , který po vyhodnocení dá stejný výsledek jako  $D$  v neomezené interpretaci s omezeným výstupem.
- (ii) je-li  $D$  výraz DRK, pak existuje doménově nezávislý výraz  $D'$ , který po vyhodnocení dá stejný výsledek jako  $D$  v omezená interpretaci.

# Sémantika DRK (6)

*Důkaz (část):* Triviálně  $\text{DRK}^{\text{rout}}$  a  $\text{DRK}^{\text{lim}}$  jsou minimálně tak silné jako  $\text{DRK}^{\text{ind}}$ , tj.  $\text{DRK}^{\text{ind}} < \text{DRK}^{\text{lim}}$  a  $\text{DRK}^{\text{rout}} < \text{DRK}^{\text{lim}}$

❖ Ukážeme sílu  $\text{DRK}^{\text{lim}}$

Je-li  $D \in \text{DRK}^{\text{ind}}$ , pak vrací  $D^{\text{dom}}[\mathbf{R}^*]$ , přičemž  $D^{\text{dom}}[\mathbf{R}^*] = D^{\text{adom}}[\mathbf{R}^*]$ .

Nechť  $D \in \text{DRK}$ . K němu lze sestrojit  $D'$  tak, že všechny volné a vázané proměnné ve formuli dotazu  $D'$  jsou omezené na aktivní doménu. Pak  $D'^{\text{adom}}[\mathbf{R}^*] = D^{\text{adom}}[\mathbf{R}^*]$ . Výraz  $D'$  je ovšem doménově nezávislý, takže  $\text{DRK}^{\text{lim}} < \text{DRK}^{\text{ind}}$ . Zároveň jsme demonstrovali část (ii) tvrzení. Platí tedy  $\text{DRK}^{\text{lim}} \cong \text{DRK}^{\text{ind}}$ .

❖ Platí, že  $\text{DRK}^{\text{rout}}$  silnější než  $\text{DRK}^{\text{lim}}$ . Důkaz (i) je technicky složitější (viz [Hull a Su 94]).

# Bezpečné formule DRK

Df.: Bezpečná formule DRK, A, je formule DRK, která je definitní a syntakticky charakterizovatelná.

1. má eliminovaný  $\forall, \Rightarrow$

2. je-li v A obsažena disjunkce, pak je podformulí

$$\varphi_1(x_1, \dots, x_s) \vee \varphi_2(x_1, \dots, x_s),$$

tj.  $\varphi_i$  obsahují stejné volné proměnné,

3. je-li v A obsažena konjunkce (maximální), např.

$\varphi \equiv \varphi_1 \wedge \dots \wedge \varphi_r$ ,  $r \geq 1$ , pak každá volná proměnná ve  $\varphi$  je omezená, tj. platí pro ni alespoň jedna

z podmínek:

- proměnná je volná ve  $\varphi_i$ , která není aritmetické porovnání a není negací,
- existuje  $\varphi_i \equiv x=a$ , kde a je konstanta,
- existuje  $\varphi_i \equiv x=y$ , kde y je omezená.

# *Bezpečné formule DRK*

4.  $\neg$  smí být použit pouze v konjunkcích typu 3.

Př.:

$x=y$  NENÍ bezpečná

$x=y \vee R(x,y)$  NENÍ bezpečná

$x=y \wedge R(x,y)$  JE bezpečná

$R(x,y,z) \wedge \neg (P(x,y) \vee Q(y,z))$  NENÍ bezpečná,  
je definitní.

$R(x,y,z) \wedge \neg P(x,y) \wedge \neg Q(y,z)$  je bezpečná!

# Ekvivalence relačních jazyků

4 přístupy:

- ❖ doménový relační kalkul (DRK)
- ❖ n-ticový relační kalkul (NRK)
- ❖ relační algebra ( $A_R$ )
- ❖ DATALOG

Dokážeme:  $DRK \cong A_R$

Lemma: Necht'  $\varphi$  je Boolský výraz vytvořený pomocí  $\neg, \wedge, \vee$  a jednoduchých selekcí  $X \theta Y$  nebo  $X \theta k$ , kde  $\theta \in \{\leq, \geq, \neq, <, >, =\}$ ,  $k$  je konstanta a  $X, Y$  jsou jména atributů. Pak k  $E(\varphi)$ , kde  $E \in A_R$  existuje relační výraz  $E'$ , jehož každá selekce je jednoduchá a  $E(\varphi) \cong E'$ .

Důkaz: 1. každá  $\neg$  se propaguje k nějaké jednoduché selekci a  $\theta$  se nahradí svou negací.

# *Ekvivalence relačních jazyků*

2. indukcí podle počtu operátorů  $\wedge, \vee$ .

Pro  $\emptyset$  operátorů - triviální

$E(\varphi) \equiv E(\varphi_1 \wedge \varphi_2)$  a  $E$  obsahuje nejvýše selekce, které jsou jednoduché. Pak  $E(\varphi) \equiv E(\varphi_1)(\varphi_2)$ .

$E(\varphi) \equiv E(\varphi_1 \vee \varphi_2)$  a  $E$  obsahuje nejvýše selekce, které jsou jednoduché. Pak  $E(\varphi) \equiv E(\varphi_1) \cup E(\varphi_2)$ .

Př.:  $E \equiv R(\neg (A_1 = A_2 \wedge (A_1 < A_3 \vee A_2 \leq A_3)) )$

pak  $\varphi \equiv A_1 \neq A_2 \vee (A_1 \geq A_3 \wedge A_2 > A_3)$

a  $E' \equiv R(A_1 \neq A_2) \cup R(A_1 \geq A_3) (A_2 > A_3)$

# Od relační algebry k DRK

Tv.: Každý dotaz vyjádřitelný v  $A_R$  je vyjádřitelný DRK.

Důkaz: indukcí podle počtu operátorů v relačním výrazu E.

1.  $\emptyset$  operátorů v E.

$$E \equiv R \rightarrow \{x_1, \dots, x_k \mid R(x_1, \dots, x_k)\}$$

$$E \equiv \text{konst. relace} \rightarrow \{x_1, \dots, x_k \mid x_1 = a_1 \wedge \dots \wedge x_k = a_k \vee x_1 = b_1 \wedge \dots \wedge x_k = b_k \vee \dots\}$$

2.  $E \equiv E_1 \cup E_2$  podle indukční hypotézy existují formule  $e_1$  a  $e_2$  s volnými proměnnými  $x_1, \dots, x_k$ ;

$$\rightarrow \{x_1, \dots, x_k \mid e_1(x_1, \dots, x_k) \vee e_2(x_1, \dots, x_k)\}$$

3.  $E \equiv E_1 - E_2$

$$\rightarrow \{x_1, \dots, x_k \mid e_1(x_1, \dots, x_k) \wedge \neg e_2(x_1, \dots, x_k)\}$$

4.  $E \equiv E_1[i_1, \dots, i_k]$

$$\rightarrow \{x_{i_1}, \dots, x_{i_k} \mid \exists x_{j_1}, \dots, x_{j(n-k)} e_1(x_1, \dots, x_n)\}$$

# *Od relační algebry k DRK*

5.  $E \equiv E_1 \times E_2$

$$\rightarrow \{x_1, \dots, x_m, x_{m+1}, \dots, x_{m+n} \mid e_1(x_1, \dots, x_m) \wedge e_2(x_{m+1}, \dots, x_{m+n})\}$$

6.  $E \equiv E_1(\varphi)$

$$\rightarrow \{x_1, \dots, x_k \mid e_1(x_1, \dots, x_k) \wedge x_A \theta x_B\}, \text{ je-li } \varphi \equiv A \theta B$$

$$\text{nebo} \quad x_A \theta a \quad \text{je-li } \varphi \equiv A \theta a$$

Dle lemmatu stačí, když  $\varphi$  označuje jednoduchou selekci.

# Sémantická definice definitních formulí

Postačující podmínky pro definitní formule  $A$ :

1. komponenty TRUE-ohodnocení  $A$  jsou z  $\text{adom}(A)$ .
2. je-li  $A' \equiv \exists y \varphi(y)$ , pak je-li pro nějaké  $y_0$   
 $\varphi(y_0) \Leftrightarrow \text{TRUE}$ , pak  $y_0 \in \text{adom}(\varphi)$ .
3. je-li  $A' \equiv \forall y \varphi(y)$ , pak je-li pro nějaké  $y_0$   
 $\varphi(y_0) \Leftrightarrow \text{FALSE}$ , pak  $y_0 \in \text{adom}(\varphi)$ .

Pz.: 2. a 3. platí pro jakékoli přípustné hodnoty volných proměnných  
 $v \varphi$  (mimo  $y$ ).

Pz.: vysvětlení podmínky 3.

$$\forall y \varphi(y) \Leftrightarrow \neg \exists y \neg \varphi(y)$$

$\Rightarrow$  je-li pro nějaké  $y_0 \neg \varphi(y_0) \Leftrightarrow \text{TRUE}$ , pak podle 2. platí  $y_0 \in \text{adom}(\neg \varphi)$ .

## Sémantická definice definitních formulí

Protože  $\text{adom}(\neg\varphi) = \text{adom}(\varphi)$ , pak

$$\varphi(y_0) \Leftrightarrow \text{FALSE} \Rightarrow y_0 \in \text{adom}(\varphi).$$

Tv.: Eliminace  $\forall$  a  $\wedge$  z definitní formule vede opět k definitní formuli.

# *Od DRK k relační algebře*

Tv.: Každý dotaz vyjádřitelný definitním výrazem DRK je vyjádřitelný v  $A_R$ .

Důkaz: indukcí podle počtu operátorů v A definitního výrazu

$$\{x_1, \dots, x_k | A(x_1, \dots, x_k)\} \quad (+)$$

- ❖  $adom(A)$  vyjádříme jako výraz  $A_R$ . Označíme ho E .
- ❖ A upravíme tak, že obsahuje pouze  $\exists, \vee, \neg$  .
- ❖ Důkaz bude pro  $adom(A)^k \cap \{x_1, \dots, x_k | A'(x_1, \dots, x_k)\}$ . Když  $A' \equiv A$  a A je definitní, vede  $\cap$  k výrazu (+).

Indukce:

1.  $\emptyset$  operátorů v  $A'$ . Pak  $A'$  je atomická formule.

$$x_1 \theta x_2 \rightarrow (E \times E)(1 \theta 2)$$

$$x_1 \theta a \rightarrow E(1 \theta a)$$

$$R(x_1, \dots, x_m) \rightarrow R(\dots \wedge i_1 \theta i_2 \wedge \dots)[\dots, i_1, \dots], \text{ bylo-li např. } x_{i1} = x_{i2}$$

# *Od DRK k relační algebře*

2. A' má alespoň jeden operátor a indukční hypotéza platí pro všechny podformule z A' s méně operátory než A'.

➤  $A'(u_1, \dots, u_m) \equiv A^1(u_1, \dots, u_n) \vee A^2(u_1, \dots, u_p)$ . Pak pro výrazy  $\text{adom}(A)^m \cap \{\underline{u}|A^i(\underline{u})\}$  existují rel. výrazy  $E_i$ . Transformace vede na  $\cup$ .

Př.:  $A'(u_1, u_2, u_3, u_4) \equiv A^1(u_1, u_3, u_4) \vee A^2(u_2, u_4)$

$\rightarrow (E_1 \times E) [1,4,2,3] \cup (E_2 \times E \times E) [3,1,4,2]$

➤  $A'(u_1, \dots, u_m) \equiv \neg A^1(u_1, \dots, u_m)$ . Pak pro výraz  $\text{adom}(A)^m \cap \{\underline{u}|A^1(\underline{u})\}$  existuje rel. výraz  $E_1$ . Transformace vede na  $-$ , tj.  $E^m - E_1$

➤  $A'(u_1, \dots, u_m) \equiv \exists u_{m+1} A^1(u_1, \dots, u_m, u_{m+1})$ . Pak pro výraz  $\text{adom}(A)^{m+1} \cap \{\underline{u}|A^1(\underline{u})\}$  existuje rel. výraz  $E_1$ . Transformace vede na  $[]$ , tj.  $E_1[1,2,\dots,m]$

Jestliže  $A' \equiv A$ , pak odpověď se nezmění.

# *Od DRK k relační algebře*

Př.:  $\{w,x \mid R(w,x) \wedge \forall y(\neg S(w,y) \wedge \neg S(x,y))\}$  je definitní výraz.

Zdůvodnění:  $\text{dom}(\neg S(w,y) \wedge \neg S(x,y)) = \text{dom}(S)$

Nechť  $y_0 \notin \text{dom}(S)$ . Pak  $\neg S(w,y_0) \wedge \neg S(x,y_0) \Leftrightarrow \text{TRUE}$ .

Tedy je splněna podmínka 3 z upřesnění definice defin. form.

Eliminací  $\wedge$  a  $\forall$  obdržíme definitní výraz:

$$\{w,x \mid \neg (\neg R(w,x) \vee \exists y(S(w,y) \vee S(x,y)))\}$$

Transformace:

$$S(w,y) \vee S(x,y) \rightarrow (S \times E)[1,3,2] \cup (S \times E)[3,1,2]$$

$$\exists y (\quad \neg \quad ) \rightarrow (\quad \neg \quad ) [1, 2] \text{ označme jako } E'$$

Pz.:  $E'$  jde optimalizovat na  $(S \times E)[1,3] \cup (S \times E)[3,1]$

$$\neg R(w,x) \rightarrow E^2 - R$$

$$\neg R(w,x) \vee \exists y(S(w,y) \vee S(x,y)) \rightarrow (E^2 - R) \cup E'$$

$$\neg (\quad \neg \quad ) \rightarrow E^2 - ((E^2 - R) \cup E')$$

# *Od DRK k relační algebře*

Problém: výsledek vede k neefektivnímu vyhodnocení

Optimalizace:

Nechť X označuje doplněk X do E. Platí:

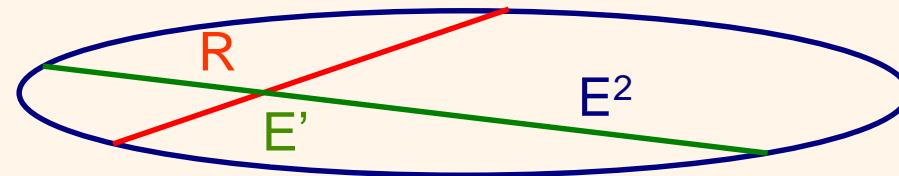
$$\underline{X \cup Y} = \underline{X} \cap \underline{Y}$$

$$\Rightarrow E^2 - ((E^2 - R) \cup E') = (E^2 - (E^2 - R)) \cap (E^2 - E')$$

=

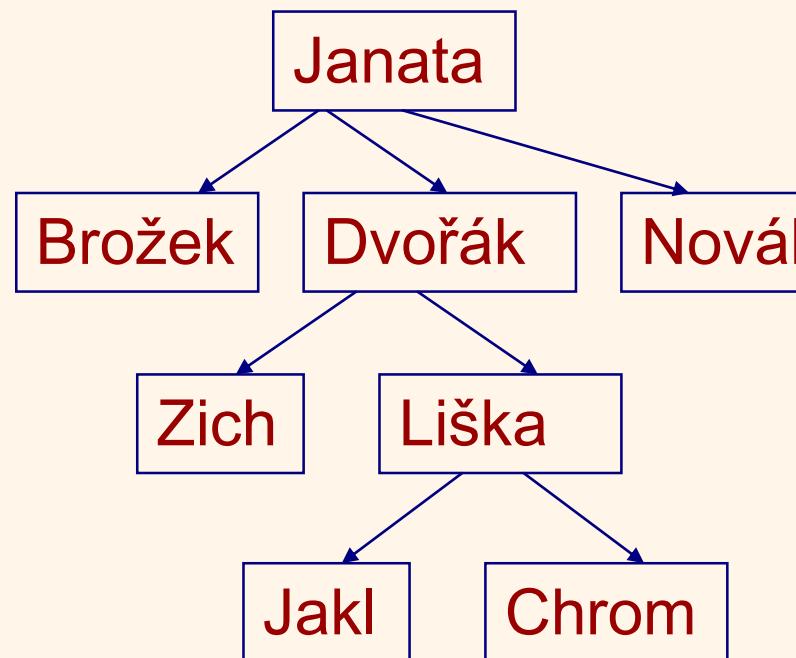
$$R \cap \underline{E'} = R - E'$$

Vizualizace:



# Vyjadřovací síla DRK ( $A_R$ )

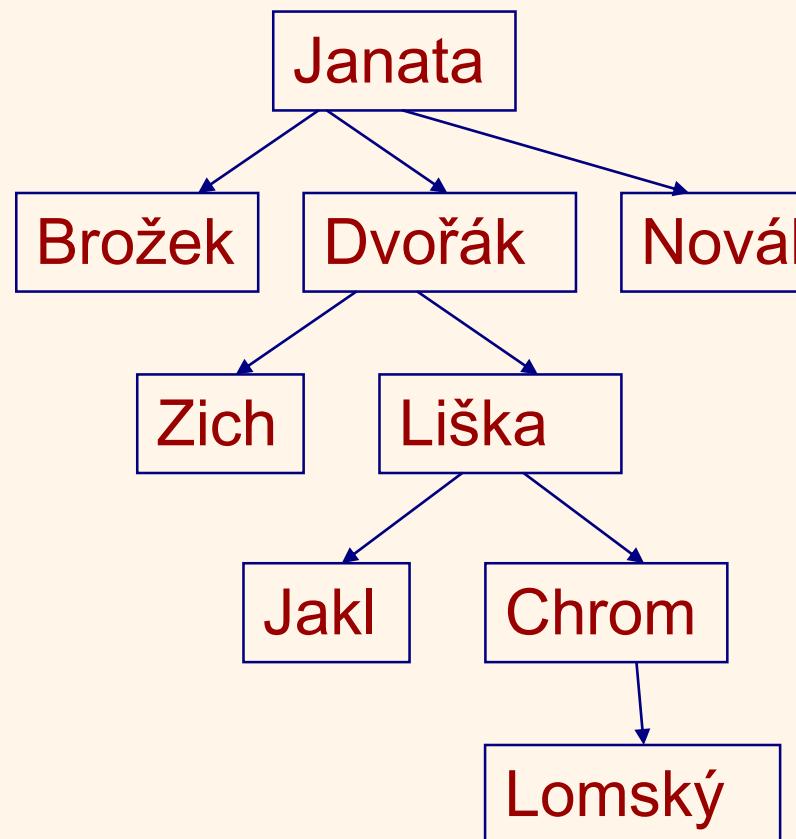
D.:Najdí všechny podřízené Dvořáka.



Pomocí:  
sjednocení  
1 spojení  
projekce

# Vyjadřovací síla DRK ( $A_R$ )

D.:Najdí všechny podřízené Dvořáka.



Pomocí  
sjednocení  
2 spojení  
projekce

# *Dotaz na tranzitivní uzávěr (0)*

Pojmy:

Df.: Binární relace  $R$  je *tranzitivní*, jestliže pro každé  $(a,b) \in R$  a  $(b,c) \in R$  také  $(a,c) \in R$ .

Df.: *Tranzitivní* uzávěr relace  $R$ ,  $R^+$ , je nejmenší tranzitivní relace obsahující  $R$ .

V databázích: schéma relace  $R$ , relace  $R^*$

Př.: NAD-POD(Nadřízený, Podřízený) vychází z tranzitivních vztahů na konceptuální úrovni.

NAD-POD\* obsahuje pouze přímé vztahy, např. (Janata, Dvořák), (Liška, Chromý), ...

Cíl: spočítat tranzitivní uzávěr relace NAD-POD\*

Předpoklad: budeme uvažovat relace, které jsou tranzitivní na konceptuální úrovni.

# Dotaz na tranzitivní uzávěr (1)

Tv.: Nechť  $R$  je schéma binární relace. Pak v  $A_R$  neexistuje výraz, který pro každou relaci  $R$  počítá její tranzitivní uzávěr  $R^+$ .

Důkaz:

1. Nechť  $\Sigma_s = \{a_1, a_2, \dots, a_s\}$ ,  $s \geq 1$ , je množina konstant, na které neexistuje uspořádání a

$$R_s = \{a_1a_2, a_2a_3, \dots, a_{s-1}a_s\}$$

Pz.:  $R_s \Leftrightarrow$  grafu  $a_1 \rightarrow a_2 \rightarrow \dots \rightarrow a_s$ , tj. transitivita je definovaná pomocí konektivity v orientovaném grafu.

Pz.: je-li na  $\Sigma_\sigma$  definováno uspořádání  $<$ , pak

$$R_s^+ \cong (R_s[1] \times R_s[2]) (1 < 2)$$

2. Ukážeme, že pro libovolný výraz  $E(R)$  existuje s takové, že  $E(R_s) \neq R_s^+$ .

# Dotaz na tranzitivní uzávěr (2)

3. Lemma: Je-li  $E$  výraz relační algebry, pak pro dostatečně velké  $s$

$$E(R_s) \cong \{b_1, \dots, b_k \mid \Gamma(b_1, \dots, b_k)\},$$

kde  $k \geq 1$  a  $\Gamma$  je formule v disjunktivní normální formě.

Atomické formule v  $\Gamma$  mají speciální tvar:

$$b_i = a_j, b_i \neq a_j,$$

$b_i = b_j + c$  nebo  $b_i \neq b_j + c$ , kde  $c$  je (ne nutně kladná) konstanta, přičemž

$b_j + c$  je zkratka pro "takové  $a_m$ , pro které  $b_j = a_{m-c}$ "

Doména interpretace pro ohodnocení proměnných  $b_j$  je  $\Sigma_s$ .

Pz.:  $b_i = b_j + c \Leftrightarrow b_i$  je za  $b_j$  ve vzdálenosti  $c$  uzelů.

# *Dotaz na tranzitivní uzávěr (3)*

## 4. Důkaz sporem.

- ❖ existuje  $E$  tak, že  $E(R) = R^+$  a jakoukoliv relaci  $R$ , tj. i  $E(R_s) = R_s^+$  pro dostatečně velká  $s$
- ❖ dle lemmatu,  $R_s^+ \cong \{b_1, b_2 \mid \Gamma(b_1, b_2)\}$

Nastávají dva případy:

(a) každá klauzule z  $\Gamma$  obsahuje atom tvaru

$$b_1 = a_i, \quad b_2 = a_i \text{ nebo } b_1 = b_2 + c \quad (\Leftrightarrow b_2 = b_1 - c)$$

Necht'  $b_1 b_2 = a_m a_{m+d}$ ,

kde  $m >$  libovolné  $i$  a  $d >$  libovolné  $c$

# Dotaz na tranzitivní uzávěr (4)

$\Rightarrow b_1 = a_m$  a  $b_2 = a_{m+d}$  nevyhovuje žádné klauzuli z  $\Gamma$ .

$\Rightarrow$  spor ( $a_m a_{m+d} \notin R_s^+$ )

(b) v  $\Gamma$  existuje klauzule s atomy obsahujícími jen  $\neq$ .

Nechť  $b_1 b_2 = a_{m+d} a_m$ , kde ani  $b_i \neq a_m$  ani  $b_i \neq a_{m+d}$   
není obsaženo v  $\Gamma$  a  $d > 0$  je větší než libovolné c

$\vee b_1 \neq b_2 + c$  nebo  $b_2 \neq b_1 + c$  v  $\Gamma$  (viz konstrukce  $\Gamma$ )

$\Rightarrow a_{m+d} a_m \in E(R_s)$  pro postačující s, ale  $\notin R_s^+ \Rightarrow$  spor

Tedy: Pro jakýkoliv výraz  $E$  vždy existuje s pro něž

$$E(R_s) \neq R_s^+$$

# *Dotaz na tranzitivní uzávěr (5)*

5. Důkaz lemmatu - indukcí dle počtu operátorů v E

I.  $\emptyset$  operátorů  $\Rightarrow E \equiv R_s$  nebo E je relační konstanta stupně 1

$\Rightarrow E \equiv \{b_1, b_2 \mid b_2 = b_1 + 1\}$ , resp.

$E \equiv \{b_1 \mid b_1 = c_1 \vee b_1 = c_2 \vee \dots \vee b_1 = c_m\}$ ,

II. a)  $E \equiv E_1 \cup E_2, E_1 - E_2, E_1 \times E_2$

$E_1 \cong \{b_1, \dots, b_k \mid \Gamma_1(b_1, \dots, b_k)\}$

$E_2 \cong \{b_1, \dots, b_m \mid \Gamma_2(b_1, \dots, b_m)\}$

$\Rightarrow$  pro  $\cup$  a - je  $k=m$  a tedy

# *Dotaz na tranzitivní uzávěr (6)*

$E \cong \{b_1, \dots, b_k \mid \Gamma_1(b_1, \dots, b_k) \vee \Gamma_2(b_1, \dots, b_k)\}$ , resp. ,

$E \cong \{b_1, \dots, b_k \mid \Gamma_1(b_1, \dots, b_k) \wedge \neg \Gamma_2(b_1, \dots, b_k)\}$ ,

$\Rightarrow$  pro  $\times$

$E \cong \{b_1, \dots, b_k, b_{k+1}, \dots, b_{k+m} \mid \Gamma_1(b_1, \dots, b_k) \wedge \Gamma_2(b_{k+1}, \dots, b_{k+m})\}$

!! pak následuje transformace do DNF.

b)  $E \cong E_1(\varphi)$  a  $\varphi$  obsahuje bud'  $=$  nebo  $\neq$

$\Rightarrow E \cong \{b_1, \dots, b_k \mid \Gamma_1(b_1, \dots, b_k) \wedge \varphi(b_1, \dots, b_k)\}$

# Dotaz na tranzitivní uzávěr (7)

c)  $E \equiv E_1[S]$

Budeme uvažovat projekci odstraňující jeden atribut

$\Rightarrow$  jde o posloupnost permutací proměnných a eliminaci poslední komponenty

eliminace  $b_k$  vede k

$\{b_1, \dots, b_{k-1} \mid \exists b_k \Gamma(b_1, \dots, b_k)\}$ , kde  $\Gamma$  je v DNF

$\Rightarrow$  podle a) ..

$\cup_{i=1 \dots m} \{b_1, \dots, b_{k-1} \mid \exists b_k \Gamma_i(b_1, \dots, b_k)\}$

$\Rightarrow$  budeme eliminovat  $\exists$  z jednoho konjunktu

❖ v  $\Gamma_i$  není  $b_k = a_i$ ,  $b_i = b_k + c$ ,  $b_k = b_i + c$

$\Rightarrow \{b_1, \dots, b_{k-1} \mid \underline{\Gamma}_i(b_1, \dots, b_{k-1})\}$

kde  $\underline{\Gamma}_i$  neobsahuje  $b_k \neq a_i$ ,  $b_i \neq b_k + c$ ,  $b_k \neq b_i + c$

# *Dotaz na tranzitivní uzávěr (8)*

# Tranzitivní uzávěr relace funkcionálně

Df.: *kompozice*  $R \circ S$  binárních relací  $R, S$  definovaných na doméně  $D$  je binární relace

$$\{a,b \mid \exists c \in D, (a,c) \in R^* \wedge (c,b) \in S^* \}$$

Nechť  $f$  je funkce přiřazující binární relaci  $R$  binární relaci  $R'$  (obě relace jsou definovány nad  $D$ ).

Df.: Nechť  $R$  je relační proměnná a  $f(R)$  relační výraz. Pak *nejmenší pevný bod* (NPB) rovnice

$$R = f(R) \tag{1}$$

je relace  $R^*$  taková, že platí:

- $R^* = f(R^*)$  /pevný bod/
- $S^* = f(S^*) \Rightarrow R^* \subseteq S^*$  /minimalita/

Df.:  $f$  je *monotónní* jestliže pro každé dvě relace  $R_1^*$ ,  $R_2^*$

$$R_1^* \subseteq R_2^* \Rightarrow f(R_1^*) \subseteq f(R_2^*)$$

# Tranzitivní uzávěr relace funkcionálně

Tv.: f je monotónní právě když platí

$$f(R_1 \cup R_2) \supseteq f(R_1) \cup f(R_2)$$

Df.: f je *aditivní* právě když platí

$$f(R_1 \cup R_2) = f(R_1) \cup f(R_2)$$

Tv.: Aditivní funkce je monotónní

Tv.(Tarski): Je-li f monotónní, pak nejmenší pevný bod rovnice (1) existuje.

*Konstrukce NPB:*

NPB se získá pro konečnou relaci R opakovanou aplikací f.

Je-li  $\emptyset$  výchozí, pak  $f^{i-1}(\emptyset) \subseteq f^i(\emptyset)$

Pak existuje  $n_0 \geq 1$  takové, že: ...

$$\emptyset \subset f(\emptyset) \subset f^1(\emptyset) \subset \dots \subset f^{n_0}(\emptyset) = f^{n_0+1}(\emptyset)$$

Relace  $f^{n_0}(\emptyset)$  je NPB rovnice (1).

# Tranzitivní uzávěr relace funkcionálně

Důkaz: indukcí podle  $i$  se ukáže, že relace  $f^{n_0}(\emptyset)$  je obsažena v každém pevném bodu rovnice (1).

Tv.: Tranzitivní uzávěr binární relace  $R^*$  definované na  $D$  je NPB rovnice

$$S = S \circ R^* \cup R^*$$

kde  $S$  je relační proměnná (binární, def. na  $D$ ).

Důkaz:  $f(S) = S \circ R^* \cup R^*$

$$\Rightarrow f^n(\emptyset) = \bigcup_{i=1..n} R^* \circ R^* \circ \dots \circ R^*$$

což vede k tranzitivnímu uzávěru

$$\bigcup_{i=1..\infty} R^* \circ R^* \circ \dots \circ R^*$$

# Tranzitivní uzávěr relace funkcionálně

Př.: Mějme schéma relace

LETADLA(Z, D, ODLET, PŘÍLET)

Úkol: vyjádřit SPOJE s přestupy

Řešení: SPOJE\* je dána jako NPB rovnice

SPOJE = LETADLA  $\cup$  (LETADLA  $\times$  SPOJE)

(2=5  $\wedge$  4<7)[1, 6, 3, 8]

Tv.: Každý výraz relační algebry neobsahující rozdíl je aditivní ve všech svých proměnných.

Pz.:

- ❖ nemonotónní výraz může mít NPB,
- ❖ je-li ve výrazu rozdíl, nemusí být výraz nemonotónní.

# *Tranzitivní uzávěr relace funkcionálně*

Df.: *Minimální pevný bod* (MPB) rovnice (1) je takový pevný bod  $R^*$ , že neexistuje žádný další její pevný bod, který je vlastní podmnožinou  $R^*$ .

$\Rightarrow \exists$  NPB, pak je jediným MPB.

Existuje-li více MPB, pak jsou navzájem neporovnatelné a NPB neexistuje.

# *Databáze intenzionálne*

Př.: mějme predikáty

$F(x,y)$   $x$  je otec  $y$

$M(x)$   $x$  je muž

$S(x,y)$   $x$  je sourozencem  $y$

$B(x,y)$   $x$  je bratr  $y$

*Extenzionální databáze (EDB):*

$F(Oldřich, Pavel)$  (1)

$F(Oldřich, Jaroslav)$  (2)

$F(Jaroslav, Veronika)$  (3)

# Databáze intenzionálne

*Intenzionální databáze (IDB):*

$M(x) :- F(x,y)$  (4)

$S(y,w) :- F(x,y), F(x,w)$  (5)

$B(x,y) :- S(x,y), M(x)$  (6)

Dotazy:

$D_1$ : Má Pavel bratra?

$D_2$ : Najdi všechny  $(x,y)$ , kde  $x$  je bratrem  $y$

$D_3$ : Najdi všechny  $(x,y)$ , kde  $x$  je sourozencem  $y$

# Řešení LP rezoluční metodou

EDB jako množina tvrzení

IDB jako množina Hornových klauzulí:

$$F(x,y) \Rightarrow M(x)$$

$$F(x,y) \wedge F(x,w) \Rightarrow S(y,w)$$

$$S(x,y) \wedge M(x) \Rightarrow B(x,y)$$

Předpoklad: formule v IDB jsou univerzálně kvantifikované,  
např.:

$$\forall x \forall y \forall w ( F(x,y) \wedge F(x,w) \Rightarrow S(y,w) )$$

Reformulace D<sub>1</sub>:  $\exists z B(z, \text{Pavel})$

# Řešení LP rezoluční metodou

## Rezoluční metoda:

- ❖ odvození sporem
- ❖ odvoditelnost je ekvivalentní odvození prázdné klauzule (NIL) v ostatních případech nelze říci, zdali je klauzule odvoditelná

Princip:  $A_1 \vee \dots \vee A_i \vee B_1 \quad C_1 \vee \dots \vee C_j \vee \neg B_2$

- ❖ pomocí *unifikace*: substitucemi se snažíme dosáhnout toho, aby  $B_1$  a  $B_2$  byly komplementární
- ❖ odvození rezolventy: je-li po unifikaci tvar vstupu  $\underline{A}_1 \vee \dots \vee \underline{A}_i \vee B \quad \underline{C}_1 \vee \dots \vee \underline{C}_j \vee \neg B$ , pak lze odvordin  $\underline{A}_1 \vee \dots \vee \underline{A}_i \vee \underline{C}_1 \vee \dots \vee \underline{C}_j$

# Řešení LP rezoluční metodou

Tv.: Rezolventa je (ne)splnitelná, byly-li vstupní klauzule (ne)splnitelné

Cíl procedury: odvodit NIL

Zdůvodnění:

$W = \{A_1, \dots, A_m\}$ , pak  $W \models A$  právě když

$A_1 \wedge \dots \wedge A_m \wedge \neg A$  je nesplnitelná

Podle Gödelovy věty je nesplnitelnost částečně rozhodnutelná, tj. existuje procedura P tak, že pro každou formuli  $\varphi$  platí:

je-li  $\varphi$  nesplnitelná,  $P(\varphi)$  se zastaví a oznámí to,

je-li  $\varphi$  splnitelná,  $P(\varphi)$  buď skončí a oznámí to, nebo neskončí.

# Řešení LP rezoluční metodou

Př.: k EDB a IDB přidáme  $\neg B(z, \text{Pavel})$  (7)

a spustíme rezoluční metodu:

- |                                     |             |
|-------------------------------------|-------------|
| (8) S(Jaroslav,w) :- F(Oldřich,w)   | $z(2),(3)$  |
| (9) S(Jaroslav,Pavel)               | $z(8),(1)$  |
| (10) M(Jaroslav)                    | $z(3),(4)$  |
| (11) B(Jaroslav,y) :- S(Jaroslav,y) | $z(10),(6)$ |
| (12) B(Jaroslav,Pavel)              | $z(11),(9)$ |
| (13) NIL                            | $z(12),(7)$ |

# Terminologie a omezení

- ❖ termy: proměnné nebo konstanty
- ❖ *tvrzení (fakty)* jsou atomické formule obsahující pouze konstanty
- ❖ *pravidla* jsou Hornovy klausule
$$L_0:- L_1, \dots, L_n$$
kde jsou atomické (positivní) formule
- ❖ atomické formule nebo negace atomických formulí se nazývají *literály*.
- ❖ *positivní a negativní literály*
- ❖ tvrzení se nazývají *základní literály*

# *Terminologie a omezení*

- ❖ struktura pravidla:

$L_0$                     *hlava pravidla*

$L_1, \dots, L_n$             *tělo pravidla*

Pz.: tvrzení i literály jsou Hornovy klauzule

# DATALOG - syntaxe a sémantika (1)

1. *(data)logický program* je množinou tvrzení a pravidel
2. tři druhy predikátových symbolů

- $R_i \in \mathcal{R}$
- $S_i \dots$  virtuální relace
- vestavěné predikáty  $\leq, \geq, \neq, <, >, =$

$R_i$  a  $S_i$  se nazývají *pravé*.

Pz.:  $\neq$  nebudeme chápat jako negaci (budeme porovnávat pouze omezené proměnné)

3. sémantiku logických programů je možné vybudovat minimálně třemi různými způsoby:

- logicko-odvozovacím přístupem,
- logicko-modelovým přístupem,
- pomocí pevného bodu zobrazení.

# DATALOG - syntaxe a sémantika (2)

## ❖ *logicko-odvozovací přístup*

Metoda: interpretace pravidel jako axiomů použitelných k důkazu, tj. provádíme substituce v těle pravidel a odvozujeme nová tvrzení z hlavy pravidel. V případě DATALOGu tak lze získat právě všechna odvoditelná tvrzení.

## ❖ *logicko-modelový přístup*

Metoda: za predikátové symboly dosadíme relace tak, aby na nich byla splněna pravidla.

Př.: Uvažujme logický program LP

IDB:       $P(x) :- Q(x)$   
               $Q(x) :- R(x),$

tj. Q a P označují virtuální relace.

# DATALOG - syntaxe a sémantika (2)

- ❖ Necht:  $\begin{array}{lll} R(1) & Q(1) & P(1) \\ & Q(2) & P(2) \\ & & P(3) \end{array}$   $M_1$

Relace  $P^*$ ,  $Q^*$ ,  $R^*$  tvoří model  $M_1$  logického programu LP.

- ❖ Necht':  $R(1)$  (a ostatní tvrzení mají hodnotu FALSE). Pak relace  $P^*$ ,  $Q^*$ ,  $R^*$  netvoří model programu P.

- ❖ Necht':  $\begin{array}{lll} R(1) & Q(1) & P(1) \end{array}$   $M_2$

Pak relace  $P^*$ ,  $Q^*$ ,  $R^*$  tvoří model  $M_2$  logického programu LP.

Necht' EDB:  $R(1)$ , tj. relační DB je dána jako

$$R^* = \{(1)\}.$$

Pak  $M_1$  i  $M_2$  jsou s danou databází konzistentní.

# DATALOG - závislostní graf (1)

- $M_2$  tvoří dokonce minimální model, tj. změníme-li v něm cokoliv, porušíme konzistenci.
- $M_1$  netvoří minimální model.

Pz.: při obou sémantikách obdržíme stejný výsledek.

Nevýhody obou přístupů: neefektivní algoritmy v případě, že EDB je dána databázovými relacemi.

❖ *pomocí pevného bodu zobrazení*

Metoda: vyhodnocovací algoritmus+relační db stroj

# DATALOG - závislostní graf (2)

Df.: **závislostní graf** logického programu P

uzly: predikáty z R a IDB

hrany: (U,V) je hrana, existuje-li pravidlo  
 $V :- \dots U \dots$

Př.: rozšíření původního příkladu

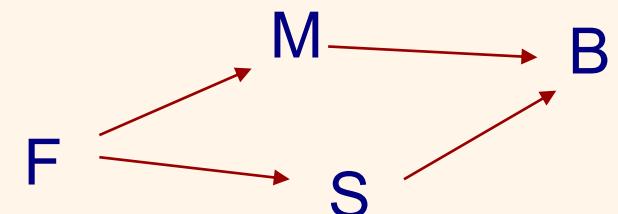
$M(x) :- F(x,y)$

$S'(y,w) :- F(x,y), F(x,w), y \neq w$

$B(x,y) :- S'(x,y), M(x)$

$C(x,y) :- F(x_1,x), F(x_2,y), S'(x_1,x_2)$

$C(x,y) :- F(x_1,x), F(x_2,y), C(x_1,x_2)$



# DATALOG - závislostní graf (3)

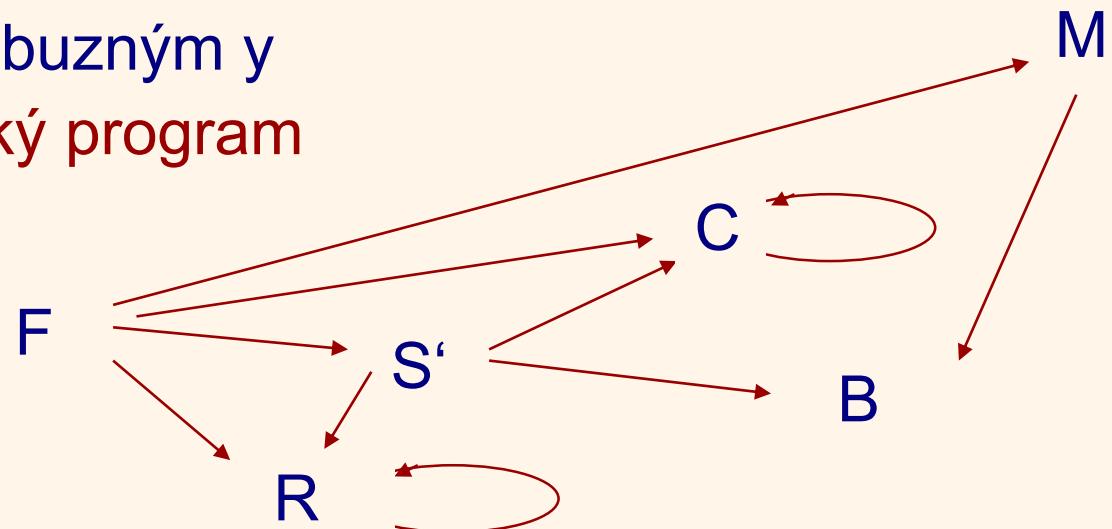
$R(x,y) :- S'(x,y)$

$R(x,y) :- R(x,z), F(z,y)$

$R(x,y) :- R(z,y), F(z,x)$

kde  $C(x,y)$  ...  $x$  je bratrem (sestřenicí)  $y$ , tj. mají otce  
bratry

$R(x,y)$  ...  $x$  je příbuzným  $y$   
rekurzivní datalogický program



# DATALOG - závislostní graf (3)

R, C ... *rekurzivní* predikáty

Df.: Logický program je *rekurzivní*, existuje-li v jeho závislostním grafu cyklus.

# DATALOG - bezpečná pravidla

Df.: *bezpečné pravidlo*

Proměnnou  $x$  vyskytující se v pravidle nazveme *omezenou*, jestliže se vyskytuje v literálu  $L$  v těle tohoto pravidla, přičemž:

- $L$  je dán pravým predikátem, nebo
- $L$  má tvar  $x = a$  nebo  $a = x$ , nebo
- $L$  má tvar  $x=y$  nebo  $y=x$  a  $y$  je omezená.

Pravidlo je *bezpečné*, jsou-li všechny jeho proměnné omezené.

Př.: bezpečnost pravidel

`JE_VĚTŠÍ_NEŽ(x,y) :- x > y`

`PŘÁTELÉ(x,y) :- M(x)`

`S'(y,w) :- F(x,y), F(x,w), y ≠ w`

# *Nerekurzivní DATALOG*

- ❖ jeho závislostní graf je acyklický
- ❖ topologické uspořádání uzelů tak, že  $R_i \rightarrow R_j$  implikuje  $i < j$

Pz.: uspořádání není dáno jednoznačně

Př.: uspořádání F - M - S - B

# Nerekurzivní DATALOG

Princip algoritmu (pro jednu virtuální relaci):

(1)  $U(x_1, \dots, x_k) :- V_1(x_{i1}, \dots, x_{ik}), \dots, V_s(x_{j1}, \dots, x_{js})$

(2) pro  $U$  se provede

převeď na spojení a selekce

projekce na výsledek

(3) kroky (1), (2) se provedou pro všechna pravidla s  $U$   
v hlavě a dílčí výsledky

se sjednotí

Pz.: vzhledem k acykličnosti a topologickému  
uspořádání lze vždy provést kroky (1), (2) na nějaké  
pravidlo

# Nerekurzivní DATALOG

Př.: konvence: proměnná  $x \rightarrow$  atribut X

Přepis pravidla

❖  $C(x,y) :- F(x_1,x), F(x_2,y), S'(x_1,x_2)$

1. krok:

$$POM(X1,X,X2,Y) = F(X1,X) * F(X2,Y) * S'(X1,X2)$$

2. krok:

$$C(X,Y) = POM[X,Y]$$

❖ pro  $S'$

$$S'(Y,W) = (F(X,Y) * F(X,W)) (Y \neq W)[Y,W]$$

# Nerekurzivní DATALOG

Další možnosti:

- ❖  $V(x,y) :- P(a,x), R(x,x,z), U(y,z)$

1. a 2. krok:

$$V(.,.) = (P(1=a)[2] * R(1=2)[1,3] * U)[.,.]$$

Problém: v hlavě pravidla mohou být konstanty, resp. stejné proměnné, různé pořadí proměnných

Požadavek na *rektifikaci*: transformace pravidel tak, aby hlavy se stejným predikátovým symbolem měly po řadě stejné proměnné

# Nerekurzivní DATALOG

Př.:  $P(a,x,y) :- R(x,y)$   
 $P(x,y,x) :- R(y,x)$

zavedeme u, v, w

substituce:

$P(u,v,w) :- R(x,y), u = a, v = x, w = y$   
 $P(u,v,w) :- R(y,x), u = x, v = y, w = x$

$\Rightarrow P(u,v,w) :- R(v,w), u = a,$   
 $P(u,v,w) :- R(v,u), w = u$

Lemma:

- (1) Je-li pravidlo bezpečné, pak po rektifikaci také.
- (2) Původní a rektifikované pravidlo je ekvivalentní, tj.  
po jeho vyhodnocení obdržíme stejnou relaci.

# *Nerekurzivní DATALOG*

Tv.: Vyhodnocený program poskytuje pro každý predikát z IDB množinu tvrzení, která tvoří

1. množinu právě těch tvrzení, dokazatelných z EDB aplikací pravidel z IDB.
2. pro EDB + IDB minimální model.

Důkaz: indukcí na pořadí pravidel.

# *Rekurzivní DATALOG*

Př.:

POD\_NAD(x,y):-PRACUJE\_PRO(x,y)

POD\_NAD(x,y):-PRACUJE\_PRO(x,z), POD\_NAD(z,y)

v EDB je relace PRACUJE\_PRO(Jméno\_z,Vedoucí)

POD\_NAD\* je tranzitivním uzávěrem relace PRACUJE\_PRO\*

Platí:

PRACUJE\_PRO  $\subseteq$  POD\_NAD

(PRACUJE\_PRO \* POD\_NAD)[1,3]  $\subseteq$  POD\_NAD

# *Rekurzivní DATALOG*

⇒ POD\_NAD\* je řešením rovnice  
$$(\text{PRACUJE\_PRO} * \text{POD\_NAD})[1,3] \cup \\ \text{PRACUJE\_PRO} = \text{POD\_NAD}$$

Obecněji:

pro IDB existuje soustava rovnic

$$E_i(P_1, \dots, P_n) = P_i \quad i=1, \dots, n$$

Řešení soustavy závisí na EDB a tvoří *pevný bod*.

Pz.: protože všechny použité operace  $A_R$  jsou aditivní,  
pevný bod existuje a dokonce nejmenší.

# Rekurzivní DATALOG

*Algoritmus: (Naivní) vyhodnocení*

*Vstup:* EDB = {R<sub>1</sub>, ..., R<sub>k</sub>}, IDB = {pravidla pro P<sub>1</sub>, ..., P<sub>n</sub>},

*Výstup:* nejmenší pevný bod P<sub>1</sub><sup>\*</sup>, ..., P<sub>n</sub><sup>\*</sup>

*Metoda:* použije se funkce eval(E) vyhodnocující relační výraz E

for i:=1 to n do P<sub>i</sub> := ∅;

repeat for i:=1 to n do

    Q<sub>i</sub> := P<sub>i</sub>;     *{ulož staré hodnoty}*

for i:=1 to n do

        P<sub>i</sub> := eval(E<sub>i</sub>(P<sub>1</sub>, ..., P<sub>n</sub>))

until P<sub>i</sub> = Q<sub>i</sub> pro všechna i ∈ <1, n>

Pz.: Jde o tzv. Gauss-Seidelovu metodu

# *Rekurzivní DATALOG*

Tv.: Vyhodnocovací algoritmus se zastaví a dá nejmenší pevný bod soustavy datalogických rovnic.

Důkaz:

- (1) plyne z toho, že *eval* je monotónní a  $P_i^*$  vznikají z konečného množství prvků.
- (2) plyne z toho, že  $P_i^*$  je řešení soustavy a navíc je součástí každého řešení pro každé  $i$ . Dokazuje se indukcí podle počtu iterací. Začíná se od  $\emptyset$ , která je součástí každého řešení.

*Nevýhody:*

- vytváření duplicitních n-tic
- vytváření zbytečně velkých relací, chceme-li ve výsledku selekci  $P_i^*$ .

# *Rekurzivní DATALOG*

## *Metoda diferencí*

Idea: v  $(k+1)$ . kroku iterace nepočítáme  $P_i^{k+1}$ , nýbrž

$$D_i^{k+1} = P_i^{k+1} - P_i^k, \text{ tj.}$$

$$P_i^{k+1} = P_i^k \cup D_i^{k+1} \text{ a tedy}$$

$$P_i^{k+1} = E_i(P_i^{k-1}) \cup E_i(D_i^k),$$

protože  $E_i$  je aditivní

Změna  $eval$  pro  $P_i$  danou jedním pravidlem:

$pincreval(E_i(\Delta P_1, \dots, \Delta P_n))$

$$= \cup_{j=1..n} eval(E_i(\dots, P_{j-1}, \Delta P_j, P_{j+1}, \dots))$$

# *Rekurzivní DATALOG*

Změna eval pro  $P_i$  danou s pravidly:

$$\begin{aligned} \text{increval}(P_k; \Delta P_1, \dots, \Delta P_n) \\ = \cup_{j=1..s} p\text{increval}(E_j(\Delta P_1, \dots, \Delta P_n)) \end{aligned}$$

Prě.:

$$\text{increval}(S') = \emptyset$$

$$\text{increval}(C) =$$

$$\begin{aligned} & (F(X1, X)^* F(X2, Y)^* \Delta S'(X1, X2))[X, Y] \cup \\ & (F(X1, X)^* F(X2, Y)^* \Delta C(X1, X2))[X, Y] \end{aligned}$$

$$\text{increval}(R) =$$

$$\begin{aligned} & \Delta S'(X, Y) \cup (\Delta R(X, Y)^* F(Z, Y))[X, Y] \cup \\ & (\Delta R(Z, Y)^* F(Z, X))[X, Y] \end{aligned}$$

# *Rekurzivní DATALOG*

## **Algoritmus: (Polonaivní)vyhodnocení**

Vstup: EDB = {R<sub>1</sub>, ..., R<sub>k</sub>}, IDB = {pravidla pro P<sub>1</sub>, ..., P<sub>n</sub>},

Výstup: nejmenší pevný bod  $P_1^*, \dots, P_n^*$

*Metoda: 1x se použije funkce eval a na difference increval*

for i:=1 to n do

$\Delta P_i := \text{eval}(E_i(\emptyset, \dots, \emptyset));$

repeat for  $i := 1$  to  $n$  do  $\Delta Q_i := \Delta P_i$ ; *{ulož staré diference}*

for i:=1 to n do begin

$$\Delta P_i := \text{increval}(E_i; (\Delta Q_1, \dots, \Delta Q_n, P_1, \dots, P_n))$$

$$\Delta P_i := \Delta P_i - P_i$$

## *{odstraň duplicity}*

end;

for  $i := 1$  to  $n$  do  $P_i := P_i \cup \Delta P_i$

until  $\Delta P_i = \emptyset$  pro všechna  $i \in \{1, n\}$

# *Rekurzivní DATALOG*

Tv.: Vyhodnocovací algoritmus se zastaví a

- ❖ dá NPB soustavy datalogických rovnic,
- ❖ NPB odpovídá právě těm tvrzením, která jsou dokazatelná z EDB pomocí pravidel z IDB.

Př.:  $R(x,y) :- P(x,y)$

$R(x,y) :- R(x,z), R(z,y)$

NPB  $R^*$  je řešením rovnice

$$R(X,Y) = P(X,Y) \cup (R(X,Z)^* R(Z,Y))[X,Y] \quad (*)$$

➤ Je-li  $P^* = \{(1,2), (2,3)\}$ , pak

$R^* = \{(1,2), (2,3), (1,3)\}$  je NPB, jehož prvky odpovídají všem odvoditelným tvrzením,

$R^*$  je i minimálním modelem.

# *Rekurzivní DATALOG*

- Je-li  $(1,1) \in R^*$ , platí  $R(1,1) :- R(1,1), R(1,1)$ , tedy i  $R^* = \{(1,1), (1,2), (2,3), (1,3)\}$  je modelem a je řešením rovnice (\*).
- Je-li  $(3,1) \in R^*$ , pak  $\{(1,2), (2,3), (1,3), (3,1)\}$  není modelem a ani není řešením rovnice.
- Nechť  $P^* = \emptyset; R^* = \{(1,2)\}$ .  
Pak  $R^*$  je modelem, ale není řešením rovnice.

# *Využití rekurzivního Datalogu ve webových službách*

Předpoklad: webové zdroje s dotazováním, které umožňuje formulovat vždy nějakou podmnožinu konjunktivních dotazů.

Př.: Amazon – zadáme jména autora a obdržíme seznam jeho knih. Dotaz na všechny dostupné knihy není možný.

Př.: Cestovní služba se zdrojovými relacemi R:  
lety(start, cíl), vlaky(start, cíl),  
autobusy(start, cíl), kyvadl\_dopr (start, cíl)

# Využití rekurzivního Datalogu ve webových službách

Datalogický program rozšiřuje možnosti konjunktivních dotazů generováním pohledů s rekurzí, např. program P

```
ans(a, b) :- lety(a,c), ind(c,b)
```

```
ind(c,b) :- lety(c,b), autobusy(b, Praha)
```

```
ind(c,b) :- lety(c,c'), ind(c',b)
```

Pz.: z P ovšem nezjistíme žádným způsobem, zdali je Praha dosažitelná odněkud někam s leteckým přestupem a dále kyvadlovou dopravou.

# Rozšíření DATALOGu o negaci

Př.:  $\text{NSR}(x,y) \dots x$  je příbuzný  $y$ , ale není sourozencem  $y$

$\text{NSR}(x,y) :- R(x,y), \neg S'(x,y)$

$\text{NSR}^* = R^* - S^{**}$

nebo

$\text{NSR}(X,Y) = R(X,Y) * \underline{S}'(X,Y)$ , kde  $\underline{S}'$  je doplněk do nějakého vhodného univerza.

Postup:

- umožníme negaci v těle pravidla, tj. negativní literály mezi  $L_1, \dots, L_n$
- bezpečná pravidla musí mít omezené proměnné, tj. zakážeme proměnné, které jsou v negativním literálu a nejsou omezené ve smyslu původní definice.

# Rozšíření DATALOGu o negaci

Problém:

ŘEŠENÍM LOGICKÉHO PROGRAMU NEMUSÍ BÝT  
NPB, ALE NĚKOLIK MPB.

Př.:     $\text{NUDNÝ}(x) :- \neg \text{ZAJÍMAVÝ}(x), \text{MUŽ}(x)$   
             $\text{ZAJÍMAVÝ}(x) :- \neg \text{NUDNÝ}(x), \text{MUŽ}(x)$

$$N(X) = M(X) - Z(X)$$

$$Z(X) = M(X) - N(X)$$

Rešení: Nechť  $M = \{\text{Honza}\}$

$M1: \{\text{NUDNÝ}^* = \{\text{Honza}\}, \text{ZAJÍMAVÝ}^* = \emptyset\}$

$M2: \{\text{ZAJÍMAVÝ}^* = \{\text{Honza}\}, \text{NUDNÝ}^* = \emptyset\}$

# *Stratifikovaný DATALOG* $\neg$

- ❖ není pravda, že jeden model je menší než druhý,
  - ❖ neexistuje žádný model menší než M1 nebo M2
- $\Rightarrow$  máme dva minimální modely

Intuice: omezení negace - použije-li se, pak na již známou relaci, tj. relace musí nejprve být definovány (event. rekurzivně) bez negace, Pak nová relace může být definována pomocí nich bez nebo s negacemi.

Df.: *Definice virtuální relace* S je množina všech pravidel, které mají S v hlavě.

Df.: S se vyskytuje v pravidlu *pozitivně (negativně)*, je-li obsažena v pozitivním (negativním) literálu.

# *Stratifikovaný DATALOG*

Df: Program P je *stratifikovateľný*, jestliže existuje dělení

$P = P_1 \cup \dots \cup P_n$  ( $P_i$  jsou navzájem disjunktní)  
takové, že pro každé  $i \in \{1, n\}$  platí:

1. Vyskytuje-li se relační symbol S pozitivně v nějakém pravidle z  $P_i$ , pak definice S je obsažena v  $\cup_{j \leq i} P_j$
2. Vyskytuje-li se relační symbol S negativně v nějakém pravidle z  $P_i$ , pak definice S je obsažena v  $\cup_{j < i} P_j$   
( $P_1$  může být  $\emptyset$ )

Df.: Dělení  $P_1, \dots, P_n$  se nazývá *stratifikace* P, každé  $P_i$  je *stratum*.

Terminologie: stratifikace ... rozvrstvení  
stratum ... vrstva

# *Stratifikovaný DATALOG*

Př.: Program  $P(x) :- \neg Q(x)$  (1)

$R(1)$  (2)

$Q(x) :- Q(x), \neg R(x)$  (3)

je stratifikovatelný. Stratifikace:  $\{(2)\} \cup \{(3)\} \cup \{(1)\}$

Program  $P(x) :- \neg Q(x)$

$Q(x) :- \neg P(x)$

není stratifikovatelný.

Df.: Nechť  $(U,V)$  je hrana závislostního grafu.  $(U,V)$  je *positivní (negativní)*, existuje-li pravidlo  $V:- \dots U \dots$  a  $U$  se tam vyskytuje pozitivně (negativně).

Pz.: Hrana může být pozitivní i negativní.

# *Stratifikovaný DATALOG* $\neg$

Tv.: Program P je stratifikovatelný právě když jeho závislostní graf neobsahuje cyklus s negativní hranou.

Důkaz:  $\Rightarrow$  Každá virtuální relace P má přiřazen index strata, ve kterém je definována. Tedy  $(P, Q)$  je pozitivní  $\Rightarrow \text{index}(P) \leq \text{index}(Q)$

$(P, Q)$  je negativní  $\Rightarrow \text{index}(P) < \text{index}(Q)$

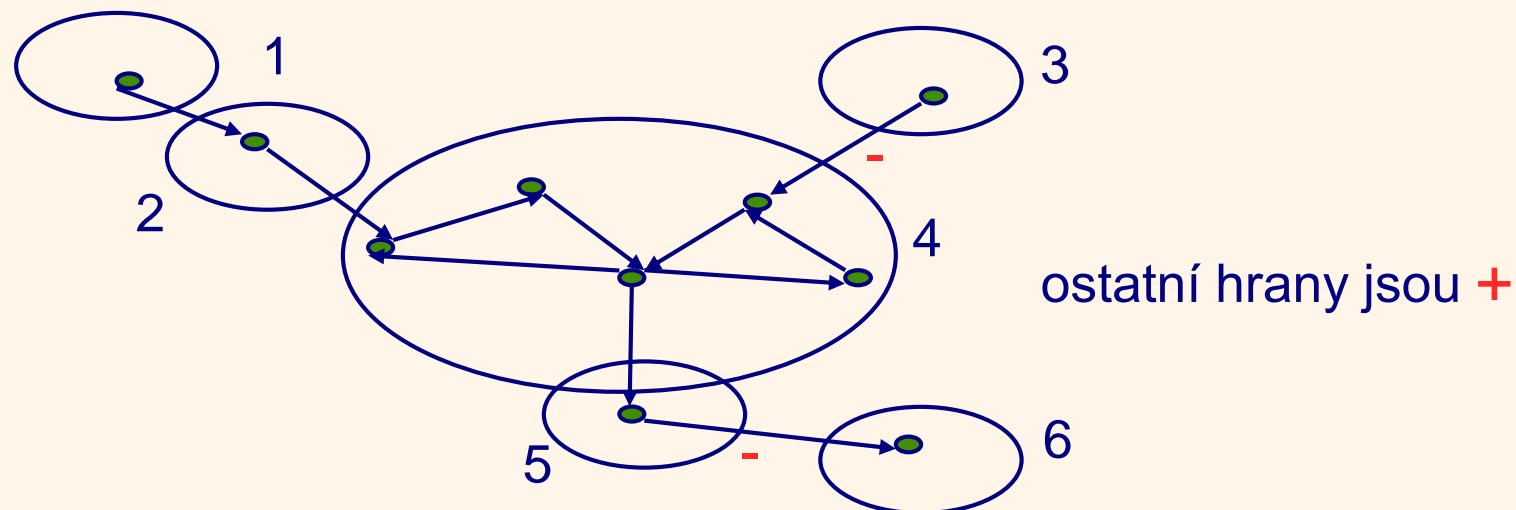
Kdyby existoval cyklus s negativní hranou, existoval by uzel X, kde  $\text{index}(X) < \text{index}(X)$ , což je spor.

$\Leftarrow$  dekomponujeme závislostní graf na silně souvislé komponenty, provedeme kondenzaci grafu, která je acyklická, a přiřadíme topologické upořádání komponentám.

# Stratifikovaný DATALOG $\neg$

Každá z komponent definuje jedno stratum, uspořádání definuje jejich očíslování. Protože negativní hrany jsou nejvýše mezi komponentami, tvoří pravidla odpovídající komponentě stratum.

Př.:



# *Stratifikovaný DATALOG*

Předpoklady: pravidla jsou bezpečná, rektifikovaná.

*adom* ... sjednocení konstant z EDB a IDB

$\neg Q(x_1, \dots, x_n)$  se transformuje na  $(adom \times \dots \times adom) - Q^*$

**Algoritmus: vyhodnocení stratifikovatelného programu**

**Vstup:** EDB = {R<sub>1</sub>, ..., R<sub>k</sub>}, IDB = {pravidla pro P<sub>1</sub>, ..., P<sub>n</sub>},

**Výstup:** minimální pevný bod P<sub>1</sub><sup>\*</sup>, ..., P<sub>n</sub><sup>\*</sup>

**Metoda:**      Najdi stratifikaci programu; spočti *adom*;

for i:=1 to s do {s strat}

begin            { pro stratum *i* existují relace spočtené ze strat *j*, kde *j*<*i*}

if ve stratu *i* je Q pozitivní then použij Q;

if ve stratu *i* je Q negativní then použij *adom*<sup>n</sup> - Q;

  použij algoritmus pro výpočet NPB

end

# *Stratifikovaný DATALOG $\neg$*

Tv.: Vyhodnocovací algoritmus se zastaví a dá MPB soustavy datalogických rovnic.

Důkaz: PB plyne indukcí podle úrovní

Pz: program stratifikovaného DATALOG $\neg$ u může mít více MPB.

# Stratifikovaný DATALOG

EDB: Díl(součástka, podsoučástka, množství)

|           |          |   |
|-----------|----------|---|
| trojkolka | kolo,    | 3 |
| trojkolka | rám      | 1 |
| rám       | sedadlo  | 1 |
| rám       | pedál,   | 2 |
| kolo      | ráfek,   | 1 |
| kolo,     | pneu     | 1 |
| pneu      | ventilek | 1 |
| pneu      | duše,    | 1 |

IDB

|  |
|--|
| Velký(P) :- Díl(P,S,Q), Q > 2          |
| Malý(P) :- Díl(P,S,Q), $\neg$ Velký(P) |

Stratifikace a výsledný MPB:

Stratum 0: Díl

Stratum 1: Velký    Velký = {trojkolka}

Stratum 2: Malý    Malý = {rám, kolo, pneumatika}

Ale: Relace Malý={trojkolka, rám, kolo, duše}, Velký={} tvoří další MPB tohoto programu, ačkoliv není výsledkem stratifikovaného vyhodnocení.

# *Stratifikovaný DATALOG*

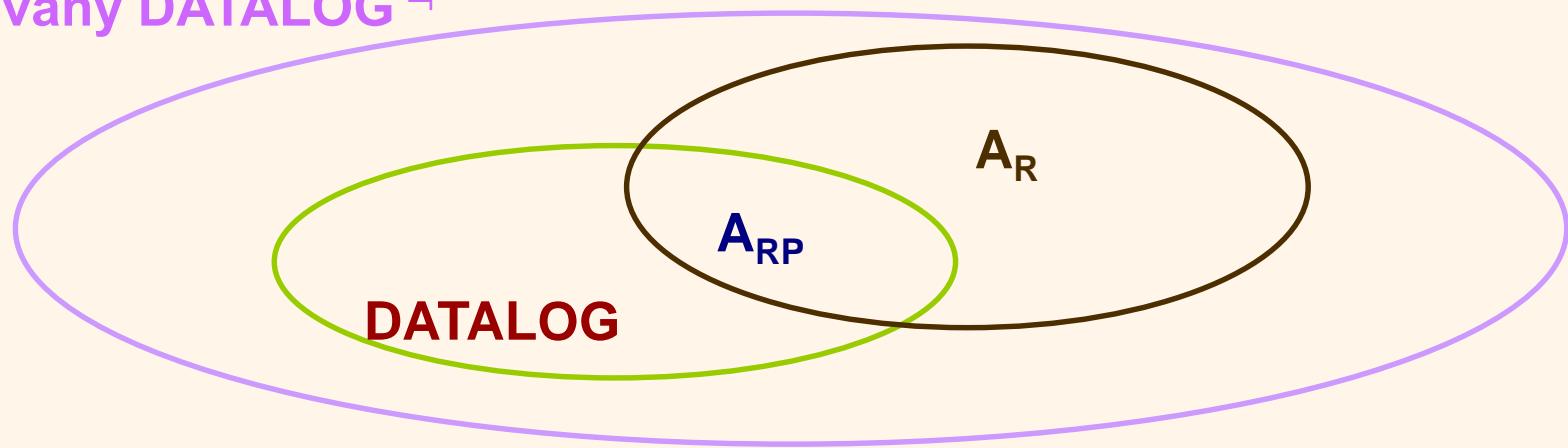
Pz.: Stratifikovatelný program má obecně více stratifikací. Ty jsou ekvivalentní, tj. vyhodnocení vede ke stejnému MPB (Apt, 1986).

Tv.: Nerekurzivní programy DATALOGu vyjadřují právě ty dotazy, které jsou vyjádřitelné monotónní podmnožinou  $A_R$ .

Pz.: *pozitivní relační algebra*  $A_{RP} \{ \times, \cup, [ ], \varphi \}$ .

# *Stratifikovaný DATALOG* $\neg$

stratifikovaný DATALOG  $\neg$



# Relační algebra a DATALOG $\neg$

Tv.: Nerekurzivní programy DATALOG $\neg$  u vyjadřují právě ty dotazy, které jsou vyjádřitelné v  $A_R$ .

Důkaz:  $\Leftarrow$  indukcí dle počtu operátorů v E

1.  $\emptyset$  operátorů:  $E \equiv R$       R je z EDB

$E \equiv$  konst. relace

pak pro každou n-tici přidej  $p(a_1, \dots, a_n)$  do EDB. Nic do IDB

2.  $E \equiv E_1 \cup E_2$

dle indukční hypotézy existují programy pro  $E_1$  a  $E_2$   
(odpovídající predikáty  $e_1$  a  $e_2$ )

$e(x_1, \dots, x_n) :- e_1(x_1, \dots, x_n)$

$e(x_1, \dots, x_n) :- e_2(x_1, \dots, x_n)$

# Relační algebra a DATALOG

3.  $E \equiv E_1 - E_2$

$e(x_1, \dots, x_n) :- e_1(x_1, \dots, x_n), \neg e_2(x_1, \dots, x_n)$

4.  $E \equiv E_1[i_1, \dots, i_k]$

$e(x_{i_1}, \dots, x_{i_k}) :- e_1(x_1, \dots, x_n),$

5.  $E \equiv E_1 \times E_2$

$e(x_1, \dots, x_{n+m}) :- e_1(x_1, \dots, x_n), e_2(x_{n+1}, \dots, x_{n+m})$

5.  $E \equiv E_1(\varphi)$

$e(x_1, \dots, x_n) :- e_1(x_1, \dots, x_n), x_{ij} = x_{ik} \text{ nebo } x_{ij} = a$

$\Rightarrow$  z nerekurzivity; topologické uspořádání + *adom<sup>n</sup>* - Q pro negaci. Pro každou P definovanou v IDB lze zkonstruovat výraz v A<sub>R</sub>. Dosazováním (podle uspořádání) obdržíme relační výrazy závisející jenom na relacích z EDB.

# Relační algebra a DATALOG

Př.: vytvoření programu z relačního výrazu

MŮŽE\_KOUPIT(X,Y) ≡

LÍBÍ\_SE(X,Y) - (DLUŽNÍK(X) × LÍBÍ\_SE(X,Y)[Y])

EDB: LÍBÍ\_SE(X,Y) osobě X se líbí předmět Y

DLUŽNÍK(X) osoba X je dlužníkem

označme DLUŽNÍK(X) × LÍBÍ\_SE(X,Y)[Y] jako  
D\_P\_PÁR(X,Y).

Pak datalogický program pro MŮŽE\_KOUPIT je:

JE\_OBDIVOVÁN(y) :- LÍBÍ\_SE(x,y)

D\_P\_PÁR(x,y) :- DLUŽNÍK(x), JE\_OBDIVOVÁN(y)

MŮŽE\_KOUPIT(x,y) :- LÍBÍ\_SE(x,y),  $\neg$  D\_P\_PÁR(x,y)

# Relační algebra a DATALOG

Př.: vytvoření relačního výrazu z programu

EDB:  $R^*, S^*$ ,  $adom \equiv R[X] \cup R[Y] \cup S$

$P(x) :- R(x,y), \neg S(y)$

$Q(z) :- S(z), \neg P(z)$

$P(X) \equiv (R(X,Y) * \{adom - S\}(Y))[X]$

$Q(Z) \equiv S(Z) * \{adom - P\}(Z) \equiv (S \cap \{adom - P\})(Z)$

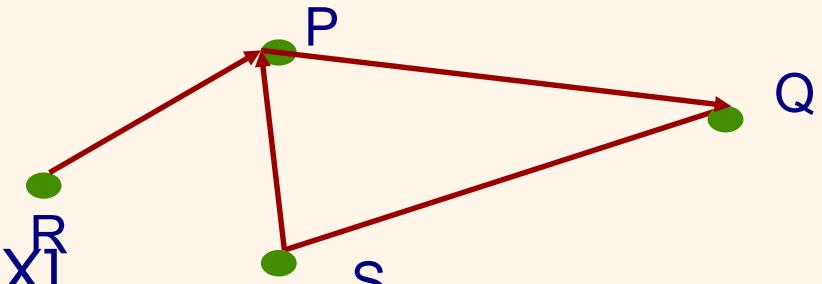
Protože  $S \subset adom$ , platí  $Q(Z) \equiv S(Z) - P(Z)$ . Po substituci za  $P$

$Q(Z) \equiv S(Z) - (R(Z,Y) * \{adom - S\}(Y))[Z]$

Pz.:  $adom$  lze nahradit  $R[Y]$

Pz.: logicky program vede k jedné výsledné relaci.

Obecněji: více (nezávislých) relací  $\Rightarrow$  více relačních výrazů



# Předpoklad uzavřeného světa (1)

Př.:  $S'(y,w) := F(x,y), F(x,w), y \neq w$

Je-li  $F^*$  taková, že nejde odvodit  $S'(Jánošík,Babinský)$ ,  
pak lze prohlásit  $\neg S'(Jánošík,Babinský)$

Pz.: Nejde o důkaz!

Df.: Uvažujme Hornovy klauzule (bez  $\neg$ ). *Předpoklad uzavřeného světa* (CWA) říká: kdykoliv tvrzení  $R(a_1, \dots, a_k)$  není odvoditelné z EDB a pravidel, pak  $\neg R(a_1, \dots, a_k)$ .

Pz.: CWA je metapravidlo k odvozování negativní informace.

Značení:  $\equiv$  CWA

# Předpoklad uzavřeného světa (2)

Předpoklady pro použití CWA:

(1) Různé konstanty neoznačují tentýž objekt

Př.:  $F(\text{Jiří}, \text{Kato})$ ,  $F(\text{Jiří}, \text{ Jánošík}) \Rightarrow S'(\text{Kato}, \text{ Jánošík})$

Jsou-li Kato a Jánošík jména téhož agenta obdržíme nesmysl

(2) Doména je uzavřená (konstanty z EDB+IDB)

Př.: jinak by nešlo odvodit  $\neg S'(\text{ Jánošík}, \text{ Babinský})$ ;  
(mohli by mít otce "mimo" databázi).

Tv.: (o konzistenci CWA): Necht'  $E$  je množina tvrzení z EDB,  $I$  je množina tvrzení odvoditelná datalogickým programem  $IDB \cup EDB$ ,  $J$  je množina tvrzení tvaru  $\neg R(a_1, \dots, a_k)$ , kde  $R$  je predikátový symbol z  $IDB \cup EDB$  a  $R(a_1, \dots, a_k)$  není v  $I \cup E$ . Pak  $I \cup E \cup J$  je logicky konzistentní.

# Předpoklad uzavřeného světa (3)

Důkaz: Nechť  $K = I \cup E \cup J$  není konzistentní.  $\Rightarrow \exists$  pravidlo  $p(\dots):-q_1(\dots), \dots, q_k(\dots)$  a substituce taková, že tvrzení na pravé straně pravidla jsou v  $K$  a odvozené tvrzení není v  $K$ . Protože tvrzení z pravé strany jsou pozitivní literály, jsou z  $I \cup E$  a ne z  $J$ . Pak ale literál z hlavy pravidla musí být z  $I$  (je odvoditelný pomocí NPB), což je spor.

Pz.: DATALOG $^{\neg}$  nelze vybudovat na základě CWA.

Př.: Uvažujme program

LP: NUDNÝ(Emil) :-  $\neg$ ZAJÍMAVÝ(Emil)

tj.  $\neg$ ZAJÍMAVÝ(Emil)  $\Rightarrow$  NUDNÝ(Emil) což je  $\Leftrightarrow$

ZAJÍMAVÝ(Emil)  $\vee$  NUDNÝ(Emil) a tedy ani

ZAJÍMAVÝ(Emil) ani NUDNÝ(Emil) nelze z LP odvodit.

# Předpoklad uzavřeného světa (4)

$LP \models_{CWA} \neg ZAJÍMAVÝ(Emil)$

$LP \models_{CWA} \neg NUDNÝ(Emil)$

Žádný model LP ale nemůže obsahovat

$\{\neg ZAJÍMAVÝ(Emil), \neg NUDNÝ(Emil)\}$

$\Rightarrow$  DATALOG $^{\neg}$  není konsistentní s CWA.

Pz.: LP má dva minimální modely:

$\{NUDNÝ(Emil)\}$  a  $ZAJÍMAVÝ\{(Emil)\}$

Stratifikace řeší příklad přirozeně:

$EDB_{LP} = \emptyset$

nejprve se spočítá  $ZAJÍMAVÝ$ , což jest  $\emptyset$ , pak  $NUDNÝ = \{Emil\}$ ,

tj. je vybrán minimální model  $\{NUDNÝ(Emil)\}$

# Předpoklad uzavřeného světa (5)

Uvažujme program

$P': \text{ZAJÍMAVÝ}(\text{Emil}) :- \neg \text{NUDNÝ}(\text{Emil})$

tj.  $\neg \text{NUDNÝ}(\text{Emil}) \Rightarrow \text{ZAJÍMAVÝ}(\text{Emil})$  což je  $\Leftrightarrow$   
 $\text{ZAJÍMAVÝ}(\text{Emil}) \vee \text{NUDNÝ}(\text{Emil})$

Stratifikace vybere model  $\{\text{ZAJÍMAVÝ}(\text{Emil})\}$

# Deduktivní databáze (1)

Neformálně: EDB  $\cup$  IDB  $\cup$  IO

Diskuse klauzulí: *klauzule* je univerzálně kvantifikovaná disjukce literálů

$$\neg L_1 \vee \neg L_2 \vee \dots \vee \neg L_k \vee K_1 \vee K_2 \vee \dots \vee K_p \quad (\Leftrightarrow)$$

$$L_1 \wedge L_2 \wedge \dots \wedge L_k \Rightarrow K_1 \vee K_2 \vee \dots \vee K_p$$

Pz.: v DATALOGu p=1

(i) k=0, p=1:

*tvrzení*, např. zam(Jiří), vydělává(Tom,8000)

*neomezené klauzule*, např. má\_rád(Bůh,x)

(ii) k=1, p=0:

*negativní tvrzení*, např.  $\neg$  vydělává(Eda,8000)

IO, např.  $\neg$  má\_rád(Jan,x)

# Deduktivní databáze (2)

(iii)  $k>1$ ,  $p=0$ :

IO, např.  $\forall x (\neg M(x) \vee \neg \check{Z}(x))$

(iv)  $k>1$ ,  $p=1$ : jde o Hornovskou klauzuli

IO nebo odvozovací pravidlo

(v)  $k=0$ ,  $p>1$ :

*disjunktivní informace*, např.  $M(x) \vee \check{Z}(x)$ ,  
 $\text{vydělává(Eda,8000)} \vee \text{vydělává(Eda,9000)}$

(vi)  $k>0$ ,  $p>1$ :

IO nebo definice neurčitých dat, např. otec  
 $\text{rodič}(x,y) \Rightarrow \text{otec}(x,y) \vee \text{matka}(x,y)$

(vii)  $k=0$ ,  $p=0$ :

prázdná klauzule (neměla by být částí databáze)

# Deduktivní databáze (3)

df.: *Definitní (určitá) deduktivní databáze* je množina klauzulí, které nejsou typu (v) a (vi). Databáze obsahující (v) nebo (vi) je *nedefinitní (neurčitá)*.

Definitní deduktivní databázi lze chápat jako dvojici

1. teorii T, která obsahuje speciální axiomy:

- tvrzení (odpovídají n-ticím z EDB)
- axiomy o prvcích:
  - úplnosti (neplatí jiná tvrzení než ta z EDB a ta odvoditelná pravidly)
  - axiom uzavřenosti domén
  - axiom jednoznačnosti jmen
- axiomy rovnosti
- množina Hornových klauzulí (*deduktivní pravidla*)

2. IO

Dotazovací jazyky

# Deduktivní databáze (4)

Pro definitní deduktivní db lze použít CWA.

Pz.: odstraní nutnost použít axiomy úplnosti a axiom jednoznačnosti jmen  $\Rightarrow$  jednodušší implementace

Tv.: Definitní deduktivní db je konzistentní.

- ❖ *odpověď na dotaz  $Q(x_1, \dots, x_k)$  v deduktivní db je množina n-tic  $(a_1, \dots, a_k)$  takových, že*  
 $T = Q(a_1, \dots, a_k),$
- ❖ deduktivní databáze *splňuje IO iff*  $\forall c \in IO \quad T = c.$

Pz.: je-li formální systém korektní a úplný, pak  $\vdash$  je totéž jako  $=$ .

# Korektnost IS (1)

DB vs. reálný svět (*svět objektů*)

Požadavky:

- ❖ konsistence
  - nelze dokázat současně  $w \sqsubset w$
- ❖ splnitelnost ve světě objektů
  - databáze je v souladu se světem objektů
- ❖ úplnost
  - v systému lze dokázat, že bud'  $w$  nebo  $\neg w$

# Korektnost IS (2)

Př.: problémy se vztahem ke světu objektů

Sch1: zam(.), plat(.), vydělává(.,.)

IO:  $\forall x (zam(x) \Rightarrow \exists y (plat(y) \wedge vyd\acute{e}lava(x,y)))$

M1: zam: {Jiří, Karel}, plat:{19500, 16700}

vydělává: { (Jiří, 19500), (Karel, 16700)},

M2: vydělává INSERT: (19500, 16700)

Sch2: zam(.), plat(.), vydělává(.,.)

IO:  $\forall x \exists y (zam(x) \Rightarrow vyd\acute{e}lava(x,y))$

$\forall x \forall y (vyd\acute{e}lava(x,y) \Rightarrow (zam(x) \wedge plat(y)))$

M2 není modelem

Dosažení konsistence: konstrukce modelu

# IO (1)

IO jako uzavřené formule.

Problémy: konzistence  
neredundantnost

Př.: funkční závislosti

❖ v jazyku logiky 1. řádu

$$\forall a,b,c_1,c_2,d_1,d_2$$

$$((R(a,b,c_1,d_1) \wedge R(a,b,c_2,d_2)) \Rightarrow c_1 = c_2))$$

❖ v teorii FZ

$$AB \rightarrow C$$

Neredundantnost se zkoumá pomocí řešení problému  
příslušnosti;

## IO (2)

### *Obecné závislosti*

$$\forall y_1, \dots, y_k \exists x_1, \dots, x_m ((A_1 \wedge \dots \wedge A_p) \Rightarrow (B_1 \wedge \dots \wedge B_q))$$

kde k, p, q  $\geq 1$ , m  $\geq 0$ ,

A<sub>i</sub> ... pozitivní literály s proměnnými z {y<sub>1</sub>, ..., y<sub>k</sub>}

B<sub>i</sub> ... rovnosti nebo pozitivní literály s  
proměnnými z {y<sub>1</sub>, ..., y<sub>k</sub>}  $\cup$  {x<sub>1</sub>, ..., x<sub>m</sub>}

m = 0 ... *plné závislosti*

# IO (3)

## Klasifikace závislostí:

- ❖ typované (1 proměnná není ve více sloupcích)
- ❖ plné, vnořené
- ❖ generující řádky, generující rovnosti
- ❖ funkční
  - inkluzní (obecně jsou vnořené, netypované)
  - šablonové ( $q=1$ , B je pozitivní literál)

# Obecné závislosti - příklad

VNOŘENÁ, GENERUJÍCÍ ŘÁDKY

$\forall x (zam(x) \Rightarrow \exists y (plat(y) \wedge vydělává(x,y)))$

PLNÁ, GENERUJÍCÍ ROVNOSTI, FUNKČNÍ

$\forall x, y_1, y_2 (vydělává(x, y_1) \wedge vydělává(x, y_2) \Rightarrow y_1 = y_2)$

PLNÁ, GENERUJÍCÍ ŘÁDKY, INKLUZNÍ

$\forall x, z (vede(x, z) \Rightarrow zam(x))$

PLNÁ OBECNĚJŠÍ

$\forall x, y, z (vydělává(x, y) \wedge vede(x, z) \Rightarrow y > 5000)$

VNOŘENÁ, GENERUJÍCÍ ŘÁDKY, INKLUZNÍ

$\forall x, z (vede(x, z) \Rightarrow \exists y (\řeší(x, y)))$

VNOŘENÁ, GENERUJÍCÍ ŘÁDKY, ŠABLONOVÁ

$\forall x, y, z ((vede(x, z) \wedge \řeší(x, y)) \Rightarrow \exists o je_\_č(x, o))$

# Tvrzení o závislostech (1)

Tv: Nejlepší procedura řešící problém příslušnosti ke třídě typovaných plných závislostí má exponenciální časovou složitost.

Pz.: Problém příslušnosti pro plné závislosti je týž pro konečné i nekonečné relace.

Př.:  $\Sigma = \{A \rightarrow B, A \subseteq B\}$

$\tau: B \subseteq A$

Platí:  $\Sigma \models_f \tau$        $\Sigma \not\models \tau$

např. na relaci  $\{(i+1, i) : i \geq 0\}$

# Tvrzení o závislostech (2)

Tv.: Problémy příslušnosti pro obecné závislosti nejsou ekvivalentní pro konečné a nekonečné relace. Oba problémy jsou neřešitelné.

Tv.: Problémy příslušnosti pro FZ a IZ je neřešitelný.

Tv.: Nechť  $\Sigma$  obsahuje pouze FZ a unární IZ. Pak problém příslušnosti pro konečné i nekonečné relace je řešitelný v polynomiálním čase.

# Tvrzení o závislostech (3)

Závěr: je-li exponenciální čas ještě únosný pro současné a budoucí počítače, jsou plné závislosti nejširší třídou upotřebitelnou pro deduktivní databáze.

⇒ významné místo Hornových klauzulí v informatice.

Pesimistický pohled:

- ❖ obecně nelze dosáhnout úplnosti
- ❖ obecně nelze dosáhnout konzistence

(vadí algoritmická složitost, kterou někdy nejde zlepšit a mnohdy ani řešit - chybí odpovídající dokazovací procedura)



# *Tvrzení o závislostech (4)*

- ❖ omezení umožní sice konzistenci, avšak odpovídající modely neodpovídají reálnému světu

Optimistický pohled:

- ❖ pesimistické výsledky jsou obecné. Jaké jsou množiny reálných závislostí?

# Problémy DJ

- ❖ 1982: Chandra a Harel postavili problém:  
Existuje dotazovací jazyk (logika), kterým lze vyjádřit právě všechny dotazy vyhodnotitelné v polynomiálním čase (PTIME)?  
Odpověď: dosud neznámá.
- ❖ 1982 Immerman, Vardi dokázali, že rozšíření logiky 1. řádu (FO) o operátor NPB (FP) to umožňuje na třídě všech upořádaných konečných struktur
- ❖ Další přiblížení: FP+C (operátor counting). Umožňuje podchytit PTIME např. na všech stromech, planárních grafech a dalších.
  - Pz.: counting umožnuje zjistit počet prvků vyhovujících formuli