Zkouška

Celkem 20 b, úspěch >=10.
Celkem 19 bodu, 12 na projiti.
celkem 25 bodu, kto mel min nez 16, tak to nedal

1. XML

Příklad:

Databáze:

FILMY(NAZ_F, DELKA, NAKLADY, REZISER, ADR)

Napište SQL/XML dotaz na: NAZ_F, DETAILY
kde atribut DETAILY má strukturu:

```
<detail delka=100>
    <reziser>Forman</reziser>
    <film>Amadeus</reziser>
</detail>
```

- hint: pomocí XMLELEMENT, XMLATTRIBUTE
- Select Naz_F, XMLELEMENT(NAME “detail”,
  XMLATTRIBUTES(Delka As “delka”),
  XMLELEMENT(NAME “reziser”, Reziser),
  XMLELEMENT(NAME “film”, Naz_f)) As Detaily
  From FILMY

2. Vyjadřovací síla

2.1 DRK,RA

Definice:

dom (doména) a **adom (akt.doména)**: pokud budu mít datový typ atributu x integer, pak dom je celý rozsah integeru a adom jsou pouze hodnoty integeru, které se nachází i v db (tj. hodnoty, kterých atribut x nabývá)

Výraz dotazu (v DRK) nazveme **doménově nezávislým (definitním, určitým)**, jestliže odpověď na něj nezávisí na dom.

- tj.: nejsou definitní pokud pod ruznymi schematy davaji ruzne vysledky

Příklad přednáška:

- ¬ KNIHA(TITUL:’Úvod do DBS’,AUTOR:a)
  - NENÍ definitní.
- ∃ič (EXEMPLÁŘ(ič,d) ∧ VÝPŮJČKA(ič,č,z))
  - JE definitní
- ∃ič (EXEMPLÁŘ(ič,d) v VÝPŮJČKA(ič,č,z))
  - NENÍ definitní, jsou-li proměnné netypované nebo s příliš “širokými” typy
    - jde o to, že pokud hledané ič, tak ho u 2. příkladu najdeme (pokud existuje), kdežto u 3. se můžeme zaseknout na neexistenci ič v EXEMPLÁŘ a už nikdy nezjistíme, jestli ič je ve VÝPUJČKA
    - viz konec tohoto dokumentu

Definice:

Proměnné opakování: volne - nejsou v zadnem kvantif. (např R(x)), vázané - jsou v nejakem kvantifikátoru (např: ∃xR(x))
Bezpečná (safe) **formule DRK**, A, je formule DRK, která je definitní a syntakticky charakterizovatelná.

1. má eliminovaný ∀
   - pozn.: ∀x φ(x) můžeme nahradit ¬∃x (¬φ(x))

2. je-li v A obsažena disjunkce φ1vφ2v..., pak φi obsahují stejné volné proměnné,
    - pozn.: φ1=>φ2 transformujeme na ¬φ1vφ2
3. je-li v A obsažena konjunkce (maximální), φ ≡ φ1∧...∧φr, r>=1, pak každá volná proměnná ve φ je **omezená** (limited), tj. platí pro ni alespoň jedna z podmínek:
    - φi není aritmetické porovnání a není negací,
        - např.: φi je ne-negovaná komplexní formule/“db predikát”
    - existuje φi ≡ x=a, kde a je konstanta,
    - existuje φi ≡ x=y, kde y je omezená.
4. ¬ smí být použita pouze v konjunkcích z bodu 3.

Příklad přednáška:

- x=y NENÍ bezpečná
    - x, y nejsou omezené
- x=y v R(x,y) NENÍ bezpečná
    - prvky disjunkce sdílí obě volné proměnné, ale první max. konjunkce (x=y) obsahuje rovnici s ne-omezenými proměnnými
- x=y ∧ R(x,y) JE bezpečná
- R(x,y,z) ∧ ¬(P(x,y) v Q(y,z)) NENÍ bezpečná, je definitní.
    - protoze disjunkce neobsahuje stejne volne promenne
- {x,y,z | R(x,y) ∧ ¬(P(x,y) v ¬Q(y,z))}
    - z neomezeno v konjunkci (volná proměnná “z” není nikde omezena) + navíc disjunkce nesdílí tytéž proměnné
- R(x,y,z) ∧ ¬P(x,y) ∧ ¬Q(y,z) je bezpečná (jedná se o ekvivalentní úpravu předchozí formule)!
    - v ¬Q(y,z) jsou sice volné y a z, ale ty jsou omezené v R(x,y,z)
    - v ¬P(x,y) jsou sice volné x a y, ale ty jsou omezené v R(x,y,z)
- ¬(x=y) ∧ R(x,y) NENÍ bezpečný
    - syntakticky se vyhodnotí jako ¬φ1∧ φ2 (tj. do této binární konjunkce vstupuje na levé straně operátor negace a na pravé straně R) a ve φ1 je maximální konjunkce x=y (a x ani y zde nejsou omezené)

Příklad:

Mame dotaz v DRK a rozhodnout, zda je/neni bezpecny
(neco jako) D(w,y)={∃x (T(x,y) ∧ (R(w,x) v S(x,y)))}

- odpoved (z fora): ano, protoze ma vsechny promenne omezene
- me prijde ze neni bezpecna viz treba pravidlo 2 (R a S nemaji stejne volne promenne)
    - není definitní
        - kvůli části R(w,x) v S(x,y)... sice při vyhodnocování už máme pevně dané x,y, ale můžeme se při vyhodnocování R zaseknout na nekonečné doméně w. To ale znamená, že kdyby se prohodilo pořadí v disjunkci na S(x,y) v R(w,x), tak už to definitní je (x, y jsou už určený z relace T).

Teorie:

Postačující podmínky pro definitní formule A:

1. komponenty TRUE-ohodnocení A jsou z *adom*(A).
2. je-li A´ ≡ ∃y φ(y), pak je-li pro nějaké y0
   φ(y0) ⇔ TRUE, pak y0 ∈ *adom*(φ).
3. je-li A´ ≡ ∀y φ(y), pak je-li pro nějaké y0
   φ(y0) ⇔ FALSE, pak y0 ∈ *adom*(φ).
    - pozn.: ∀y φ(y) ⇔ ¬∃y ¬φ(y)

Příklad přednáška:

převod z DRK na RA

{w,x | R(w,x) ∧ ∀y(¬S(w,y) ∧ ¬S(x,y))} je definitní výraz.
Zdůvodnění: dom(¬S(w,y) ∧ ¬S(x,y)) = dom(S)

Nechť y0 ∉ dom(S). Pak ¬S(w,y0) ∧ ¬S(x,y0) ⇔ TRUE.
Tedy je splněna podmínka 3 z upřesnění definice defin. form.

Eliminací ∧ a ∀ obdržíme definitní výraz:
{w,x | ¬(¬R(w,x) ∨ ∃y(S(w,y) ∨ S(x,y))}

Transformace:
S(w,y) ∨ S(x,y) → (SxE)[1,3,2] ∪ (SxE)[3,1,2]
∃y (-”-) → (-”-) [1, 2] označme jako E’
Pz.: E’ jde optimalizovat na (SxE)[1,3] ∪ (SxE)[3,1]
¬R(w,x) → E^2 - R
¬R(w,x) ∨ ∃y(S(w,y) ∨ S(x,y) → (E^2 - R) ∪ E’
¬(-”-) → E^2 -((E^2 - R) ∪ E’)

Příklad:

Máme relace T(X,W), S(X,Y) a R(A,B) a dotaz v DRK
{y,w | ∃ x (T(x,w) ∧ (S(x,y) ∨ R(x,y)))}

-
- je bezpečný?
  - je bezpečná (volné y,w jsou omezené)
- Prevedte pomoci algoritmu do RA (2 body)
  - *forum říká: algoritmem se myslelo nejspis to, co je popsane na slidech (ty kde je i datalog) - napr. formule mela byt nejdriv normalizovana...*
  - Definitni:
    - Necht’ z patri adom(T v S v R) potom T(z,w) & (S(z,y) ∨ R(z,y)) ⇔ TRUE
      - 2) podminka splnena
  - Normalizace (eliminace ∧ a ∀):
    - {y,w | ∃ x ¬¬(T(x,w) ∧ (S(x,y) ∨ R(x,y)))}
    - {y,w | ∃ x ¬(¬T(x,w) ∨ ¬(S(x,y) ∨ R(x,y)))}
  - Transformace:
    - 3) S(x,y) ∨ R(x,y) -> S ∪ R … E’
    - 4) ¬E’ -> E^2 – E’
    - 5) ¬T(x,w) -> E^2 – T
    - 6) E^2 – T ∨ E^2 – E’ -> (E^2 – T x E) [2,3] ∪ (E^2 – E’ x E) [3,2]
    - 6) ¬(¬T(x,w) ∨ ¬(S(x,y) ∨ R(x,y))) -> E^3 – ((E^2 – T x E) [2,3] ∪ (E^2 – E’ x E) [3,2])
    - 7) ∃x -> E^3 – ((E^2 – T x E) [2,3] ∪ (E^2 – E’ x E) [3,2]) [2,3]
    - 8) E^3 – ((E^2 – T x E) [2,3] ∪ (E^2 – (S ∪ R) x E) [3,2]) [2,3]

Příklad:

Máme schéma KNIHA(název, autor, isbn) a EXEMPLÁŘ(isbn, cena, země, ...)

- Napište pohled v SQL DRAHEKNIHY(nazev, autor) - knihy s cenou na 4000 a zemí původu Velká Británie, Německo nebo Francie.
  - CREATE VIEW DRAHEKNIHY(nazev, autor) AS
    SELECT k.nazev, k.autor
    FROM KNIHA k, EXEMPLÁŘ e
    WHERE
    k.isbn = e.isbn AND
    e.cena > 4000 AND
    e.zeme IN (‘Velká Británie’,’Německo’,’Francie’)
    GROUP BY e.isbn
- Převědte do RA (1 bod)
  - (KNIHA*EXEMPLÁŘ)(E.CENA > 3 ∧ (E.ZEME = ‘Velká Británie’ ∨...))[K.NAZEV,K.AUTOR]
- Převědte a) do DRK pomocí algoritmu (1 bod)
  - bez algoritmu:
    - {n, a | ∃i, c, z KNIHA(n,a,i) ∧ EXEMPLÁŘ(i,c,z) ∧ c > 4000 ∧ (z = ‘Velká Británie’ ∨...) }

- výsledek RA i DRK dotazu je množina (=automatický DISTINCT)

Příklad:

mame DB: Oddeleni(id_oddeleni, jmeno_oddeleni, id_vedouciho),
Zamestnanec(jmeno,id_zamestnance,jmeno_oddeleni)
- jmena oddeleni se mohla opakovat
- id vedouciho odkazuje do zamestancu

- napsat view na : vybrat jmena oddeleni a jejich vedouci tech oddeleni, jejichz nazev se v databazi neopakuje
  - CREATE VIEW Oddeleni_Vedouci AS
    ```
    SELECT o.jmeno_oddeleni, MAX(z.jmeno) AS jmeno
    FROM Oddeleni o, Zamestnanec z
    WHERE
        o.id_vedouciho = z.id_zamestnance
    GROUP BY o.jmeno_oddeleni
    HAVING COUNT(*) = 1
    ```
  - CREATE VIEW Oddeleni_Vedouci AS
    ```
    SELECT o.jmeno_oddeleni, z.jmeno
    FROM Oddeleni o, Zamestnanec z
    WHERE
        o.id_vedouciho = z.id_zamestnance AND
        o.jmeno_oddeleni NOT IN (
            SELECT jmeno_oddeleni FROM Oddeleni
            GROUP BY jmeno_oddeleni HAVING COUNT(*)>1)
    ```
- rozhodnout, zda je to view, co jsme napsali v 1. aktualizovatelny
  - nejde protože používáme GROUP BY (subquery taky vadí :))
  - SQL92:
    - V *dotazu* nesmí být klauzule **DISTINCT**, tj. nesmí být eliminovány duplicitní záznamy.
    - Každý sloupec *dotazu* musí být přímo sloupcem tabulky, tj. nesmí to být ani výraz ani agregační funkce a každý sloupec tabulky smí být vybrán jen jednou.
    - Data *dotazu* nesmí být řazena, tj. není povoleno použití **ORDER BY**.
    - *Dotaz* nesmí být hierarchický, tj. nesmí obsahovat klauzule **CONNECT BY** a **START WIDTH**.
    - Pohled je specifikován nad jedinou tabulkou a v případě, že je nad pohledem, musí tento pohled splňovat všechny tyto podmínky. To znamená, že pohled nesmí být spojením více tabulek ani jejich sjednocením, průnikem či rozdílem (**UNION**, **INTERSECT**, **MINUS**).
    - Klauzule **WHERE** *dotazu* nesmí obsahovat vnořený příkaz **SELECT**.
    - V *dotazu* nesmí být použita klauzule **GROUP BY** včetně **HAVING**.
    - Zdroj: http://www.kiv.zcu.cz/~zima/vyuka/db2/sql92-03.html
  - SQL99: Columns (of views) are potentially updatable if …
    - no DISTINCT operator
    - no GROUP BY, HAVING clause
    - no derived columns (e.g. arithmetic expressions)
- napsat dotaz view v DRK
  - $\{x, y \mid Oddeleni(io, x, iv) \wedge Zamestnanec(y, iv, jo) \wedge \neg\exists io2(Oddeleni(io2, x, iv2) \wedge \neg(io=io2))\}$
    - formálně by se ještě mělo zapsat existenční kvantifikátory pro: io, iv, jo, iv2:
      $\{x, y \mid \exists\ io, iv, jo\ ($
      $Oddeleni(io, x, iv) \wedge$
      $Zamestnanec(y, iv, jo) \wedge$
      $\neg\exists\ io2, iv2\ (Oddeleni(io2, x, iv2) \wedge \neg(io=io2))\}$

Další příklady:

1b - napsat nějakej pohled (vznosně to bylo nazváno virtuální relace)
1b - převést do RA (algoritmem!!!)
1b - převést do DRK (algoritmem!!!)

2b - napsat nějakej DRK (taky algoritmem!!!)

1b - estli něco jako metro(linka, Můstek, Skalka) podle programu implikuje (linka, Skalka, Můstek) (neimplikovalo)
2b - kolik relací z programu de přepsat do RA

2.2 Datalog

Teorie:

Extenzionální databáze (EDB) - db predikáty
Intenzionální databáze (IDB) - pravidla
Tvrzení: Nerekurzivní programy DATALOG¬u vyjadřují právě ty dotazy, které jsou vyjádřitelné v A_R.

Příklad z přednášky:

vytvoření programu z relačního výrazu
MŮŽE_KOUPIT(X,Y) = LÍBÍ_SE(X,Y) - (DLUŽNÍK(X) x LÍBÍ_SE(X,Y)[Y])

- "může si koupit ten X, komu se líbí nějaké Y, mínus ty případy, kdy X je zadlužen"

EDB: LÍBÍ_SE(X,Y) osobě X se líbí předmět Y
DLUŽNÍK(X) osoba X je dlužníkem
označme DLUŽNÍK(X) x LÍBÍ_SE(X,Y)[Y] jako D_P_PÁR(X,Y).

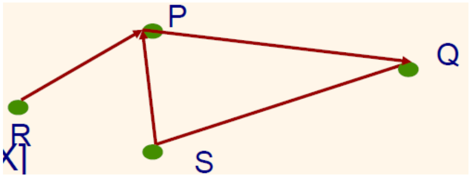
Pak datalogický program pro MŮŽE_KOUPIT je:
JE_OBDIVOVÁN(y) :- LÍBÍ_SE(x,y)
D_P_PÁR(x,y) :- DLUŽNÍK(x), JE_OBDIVOVÁN(y)
MŮŽE_KOUPIT(x,y) :- LÍBÍ_SE(x,y), ¬D_P_PÁR(x,y)

Příklad z přednášky:

vytvoření relačního výrazu z programu
EDB: R*, S*, adom = R[X] ∪ R[Y] ∪ S
P(x) :- R(x,y), ¬S(y)
Q(z) :- S(z), ¬P(z)

P(X) = (R(X,Y) * {adom - S}(Y))[X]
Q(Z) = S(Z) * {adom - P}(Z) (S ∩ {adom - P})(Z)

Protože S ⊂ adom, platí Q(Z) = S(Z) - P(Z). Po substituci za P
Q(Z) = S(Z) - (R(Z,Y) * {adom - S}(Y))[Z]



Teorie:

Program P je **stratifikovatelný**, jestliže existuje dělení P = P1 ∪ … ∪ Pn (Pi jsou navzájem disjunktní) takové, že pro každé i ∊ <1,n> platí:

1. Vyskytuje-li se relační symbol S pozitivně v nějakém pravidle z Pi, pak definice S je obsažena v $\cup_{j<=i}$ Pj
2. Vyskytuje-li se relační symbol S negativně v nějakém pravidle z Pi, pak definice S je obsažena v $\cup_{j<i}$ Pj

(P1 může být ∅)

Dělení P1,…, Pn se nazývá **stratifikace** P, každé Pi je **stratum**.

Příklad z přednášky:

Program P(x) :- ¬Q(x) (1)
R(1) (2)
Q(x) :- Q(x), ¬R(x) (3)

je stratifikovatelný. Stratifikace: {(2)} ∪ {(3)} ∪ {(1)}

Program P(x) :- ¬Q(x)
Q(x) :- ¬P(x)

není stratifikovatelný.

Příklad z knížky (Abiteboul, Hull: Foundations of Databases):

Consider a database for the Parisian Metro. Note that this database essentially describes a graph. (Database applications in which part of the data is a graph are common.) To avoid making the Metro database too static, we assume that the database is describing the available metro connections on a day of strike (not an unusual occurrence). So some connections may be missing, and the graph may be partitioned.

EDB(Pmetro) = {Links},
IDB(Pmetro) = {St_Reachable, Li_Reachable, Ans_1, Ans_2, Ans_3}

St_Reachable(x, x) ←
St_Reachable(x, y) ← St_Reachable(x, z), Links(u, z, y)
Li_Reachable(x, u) ← St_Reachable(x, z), Links(u, z, y) *//dosažitelná trasa*
Ans_1(y) ← St_Reachable(Odeon, y)
Ans_2(u) ← Li_Reachable(Odeon, u)
Ans_3() ← St_Reachable(Odeon, Chatelet)

info:

1. What are the stations reachable from Odeon?
2. What lines can be reached from Odeon?
3. Can we go from Odeon to Chatelet?

For example, an instantiation of the second rule of Pmetro is as follows:

St_Reachable(Odeon, Louvre)←St_Reachable(Odeon, Chatelet), Links(1, Chatelet, Louvre)

Příklad:

DATALOG. Mame METRO(linka, stanice, nasledujici stanice)

A(x,x):-METRO(u,x,y) (1)
A(x,y):-A(x,z),METRO(u,z,y) (2)
B(x,z):-A(x,y),A(z,y),x!=z (3)
O1(y):-A(y,Skalka),y!=Krizikova (4)
O2(z):-B(z,Krizikova) (5)

- Co znamenaji A, B, O1 a O2? (4 body)
  - (1) (2) - A říká rekurzí jestli je stanice dosažitelná
  - (3) - B(x,z) - stanice, ze kterých je dostupná nějaká společná stanice
  - O1 - stanice z kterých je dosazitelná Skalka mimo Křižíkovy
  - O2 - stanice, pro kterou existuje společně dostupná stanice s Křižíkovou
- je A(x,x) true i pro stanice, které existují, ale nemají následníka? jako jednosměrná konečná
  - není, ale původní zadání v knížce to umožňovalo
- jak upravit program aby fungoval pro stanice se sejným jménem (např: Národní divadlo, existují dvě)
  - unikátní index? :)
- je program stratifikovatelný?
  - je, stratifikace: {(1)} ∪ {(2)} ∪ {(3)} ∪ {(4)} ∪ {(5)}

        - nakonec jsme se shodli že METRO je fakt :))
        - kdyz vezmem orig. příklad (viz vejš) tak METRO patří do EDB
        - ak existuje zavislostny graf ktory neobsahuje cyklus s negativnou hranou tak je stratifikovatelny (ked si nakreslim ten graf tak tam ziadna negativna hrana nieje + jediny cyklus je A -> A)
- Ktery z A,B,O1 a O2 lze spocitat v RA? (2 body)
    - musí to být nerekurzivní DATALOG¬
        - A je cyklická => rekurzivní
        - O1,B je def. pomocí A
        - O2 je def. pomocí B
            - => ani jedno nelze spočítat v RA
- v databazi je (A, Mustek, Skalka), lze z toho odvodit A(Skalka, Mustek) ?
    - ne - predikát A popisuje jednosměrnou dosažitelnost

Příklad:

mame DB: Oddeleni(id_oddeleni, jmeno_oddeleni, id_vedouciho),
Zamestnanec(jmeno,id_zamestnance,jmeno_oddeleni)
- jmena oddeleni se mohla opakovat
- id vedouciho odkazuje do zamestancu

napsat v DATALOGu:

- to same co 1., tj: vybrat jmena oddeleni a jejich vedouci tech oddeleni, jejichz nazev se v databazi neopakuje
    - JineOddeleni(io, x) :- Oddeleni(io2, x, iv2), io!=io2
      Oddeleni_Vedouci(x,y) :- Oddeleni(io, x, iv), Zamestnanec(y, iv, jo), ¬JineOddeleni(io, x)
- vybrat zamestnance, kteri pracuji soucasne v oddeleni TV a oddeleni HUDBA
    - TVHUDBA(x) :- zamestnanec(x, _, TV), zamestnanec(x, _, HUDBA)
- vybrat zamestance, kteri pracuji bud v oddeleni OBUV nebo OBLEKY
    - OBUVOBLEKY(x) :- zamestnanec(x, _, OBUV)
      OBUVOBLEKY(x) :- zamestnanec(x, _, OBLEKY)
- vybrat zamestnance, kteri nepracuji v oddeleni : TV, HUDBA, OBUV, OBLEKY
    - NEPRACUJI(id) :-
      Zamestnanec(id, _, _), ¬Zamestnanec(id, _, HUDBA), ¬Zamestnanec(id, _, OBLEKY)...

- udelat dotaz, ktery vypise vsech vedoucich (funkce mela byt tvaru NADRIZENY(ZAMESTANEC, VEDOUCI))
    - NADRIZENY(ZAMESTANEC, VEDOUCI):-
      Zamestnanec(ZAMESTANEC, _, x),
      Oddeleni(_, x, y),
      Zamestnanec(VEDOUCI, y, x)
- vypsat top-vedoucich, kteri nemaji nikoho nadrazeneho (zrejmne se tam melo pouzit funkce NADRIZENY(ZAMESTANEC, VEDOUCI), pac to bylo za 0.5 bodu)
    - TOPVEDOUCI(VEDOUCI) :- Nadrizeny(_, VEDOUCI), ¬Nadrizeny(VEDOUCI, _)
        - jenom NADRIZENY(_, VEDOUCI) nestačí - vrátí jenom všechny nadřízené
            - kdy může být vedoucí nad vedoucím - např v případě kdy vedoucí oddělení pracuje jako zaměstnanec jiného oddělení. Nebo když jsou oddělení hierarchicky dělená a vedoucí nižších oddělení jsou zaměstnanci vyšších oddělení...

3. Rekurze v SQL

Příklad z přednášky:

Tabulka: Zaměstnanci(č_zam, jméno, funkce, č_nad)
Najdi všechny nadřízené Nového (včetně něho sama)

```
WITH RECURSIVE Nadřízení(jméno, č_nad, č_zam) AS
    (SELECT jméno, č_nad, č_zam
```

```
                FROM Zaměstnanci
                WHERE jméno = 'Nový'
                UNION ALL
        SELECT Z.jméno, Z.č_nad, Z.č_zam
                FROM Zaměstnanci AS Z
                        INNER JOIN
                        Nadřízení AS N
                        ON N.č_nad = Z.č_zam)
SELECT * FROM Nadřízení
```

Příklad:

Rekurzivní vyhledávání: lety z přednášky

Příklad:

Mame tabulku Zamestnanci(jmeno, plat, vedouci). Najdete pomoci rekurzivniho dotazu vsechny zamestnance s platem nad 100 000, kteri jsou (i neprimi) podrizeni Ryby. (3 body)

```
WITH RECURSIVE PodRybou(jméno) AS
        (SELECT jméno
        FROM Zamestnanci
        WHERE vedouci = “Ryba”
UNION ALL
        SELECT jméno
        FROM Zaměstnanci Z, PodRybou P
        WHERE Z.vedouci = P.jmeno
)
SELECT * FROM PodRybou
WHERE plat > 100 000
```

4. Herbrandovské struktury a báze, svazy, produkční operátor

4.1 Produkční operátor T_P

Příklad z wiki:

- Opáčko
  $T_P\uparrow 0 = \{\}$
  $T_P\uparrow k = T_P(T_P\uparrow k-1)$
  $T_P\uparrow \alpha = \cup\{ T_P\uparrow \beta: \beta<\alpha \}$ .. α limitní
  $T_P\downarrow 0 = B_P$
  $T_P\downarrow k = T_P(T_P\downarrow k-1)$
  $T_P\downarrow \alpha = \cap\{ T_P\downarrow \beta: \beta<\alpha \}$ .. α limitní
- Příklad
  P = {
  q(b) <−
  q(f(x)) <− q(x)
  p(f(x)) <− p(x)
  p(a) <− p(x)
  r(c) <− r(x), q(x)
  r(f(x)) <− r(x)
  }
- Spočtěte **lfp**
  $T\uparrow 0 = \{\}$
  $T\uparrow 1 = \{ q(b) \}$ .. *přidáme prvky, co jsou důsledkem předchozího kroku*
  $T\uparrow 2 = \{ q(b), q(f(b)) \}$
  ..
  $T\uparrow k = \{ q(b), q(f(b)), .., q(f^{k-1}(b)) \}$
  ..
  $T\uparrow \omega = \{ q(b), q(f(b)), .., q(f^n(b)), .. \} = lfp$ .. *dál se již nemění*
- Spočtěte **gfp**
  Herbrandovská báze $B_P = \{ p(X), p(f^n(X)), q(X), q(f^n(X)), r(X), r(f^n(X)); X\in\{a,b,c\}, n>=1 \}$
  $T\downarrow 0 = B_P$
  $T\downarrow 1 = \{ p(a), p(f^m(a)), p(f^n(b)), p(f^n(c)), q(b), q(f^m(b))\ q(f^n(a)), q(f^n(c)), r(c), r(f^m(c)), r(f^n(a)), r(f^n(b));$ $n>=2, m>=1 \}$ .. *odstraníme prvky, co nejsou důsledkem předchozího kroku*
  ..
  $T\downarrow k = \{ p(a), p(f^m(a)), p(f^n(b)), p(f^n(c)), q(b), q(f^m(b))\ q(f^n(a)), q(f^n(c)), r(c), r(f^m(c)), r(f^n(a)), r(f^n(b));$ $n>=k+1, m>=1 \}$
  ..
  $T\downarrow \omega = \{ p(a), p(f^m(a)), q(b), q(f^m(b)), r(c), r(f^m(c)); m>=1 \}$ .. *není to ještě fix-point*
  $T\downarrow \omega+1 = \{ p(a), p(f^m(a)), q(b), q(f^m(b)), r(f^m(c)); m>=1 \}$
  ..
  $T\downarrow \omega*2 = \{ q(b), q(f^m(b)), p(a), p(f^m(a)); m>=1 \} = gfp$ .. *dál se již nemění*

5. Statická analýza dotazovacích jazyků

5.1 Tableau dotazy

Příklad z přednášky:

Převeďte tablo na Datalog dotaz:

A	B
a_1	b_2
b_1	a_2
b_1	b_2
a_1	a_2

- u(a1; a2) :- AB(a1, b2), AB(b1, a2), AB(b1, b2)

Příklad:

relace vypadaly takto: R(A,B,C), S(C,D,E)

- napsat ekvivalentni tablo dotaz k RA dotazu (R*S)(C=4)[A,B]
  - q = (T,u)
    - T = R(x_A,x_B,4), S(4,x_D,x_E)
    - u = <A:x_A, B:x_B>
- napsat ekvivalentni tablo dotaz k RA dotazu (R*S)(B = 3)[A,E]
  - q = (T,u)
    - T = R(x_A,3,x_C), S(x_C,x_D,x_E)
    - u = <A:x_A, E:x_E>
- napsat ekvivalentni tablo dotaz DRK dotazu: {x,y,z| R(x,2,z) ∧ R(y,3,x) ∧ R(z,y,1) }
  - q = (T,u)
    - T = R(x,2,z), R(y,3,x), R(z,y,1)
    - u = <x, y, z>
  - zapis **T** tabulkou podle me:

R	R.A	R.B	R.C
	x	2	z
	y	3	x
	z	y	1

- napsat ekvivalenti tablo dotaz k DATALOG dotazu : dotaz(x,y):- R(1,x,z),R(x,2,y)
  - q = (T,u)
    - T = R(1,x,z), R(x,2,y)
    - u = <x, y>

Příklad:

R(A,B,C), S(C,D,E)

Dotaz v RA ((R*S)(B=2))[A,E] (2 body)

- prevest dotaz v RA na tabulkovy dotaz
  - q = (T,u)
    - T = R(x_A,2,x_C), S(x_C,x_D,x_E)
    - u = <A:x_A, E:x_E>

Příklad:

tableau dotaz **T:**

R	A	B	C
	x	z	v

S	D	A	E
	y	x	3

Q	D	F
	y	2

u = <A:x,D:y,B:z>

- Ekvivalentni dotaz v DATALOGU (1 bod)
  - u(x,y,z) ← R(x,z,v), S(y,x,3), Q(y,2)
- Ekvivalentni dotaz v RA (1 bod)
  - (R*S*Q)(E = 3 ∧ F = 2)[A,D,B]
- Ekvivalentni dotaz v DRK (1 bod)
  - {x,y,z| R(x,z,v) ∧ S(y,x,3) ∧ Q(y,2) }

5.1.1 Homomorfismus tableau dotazů

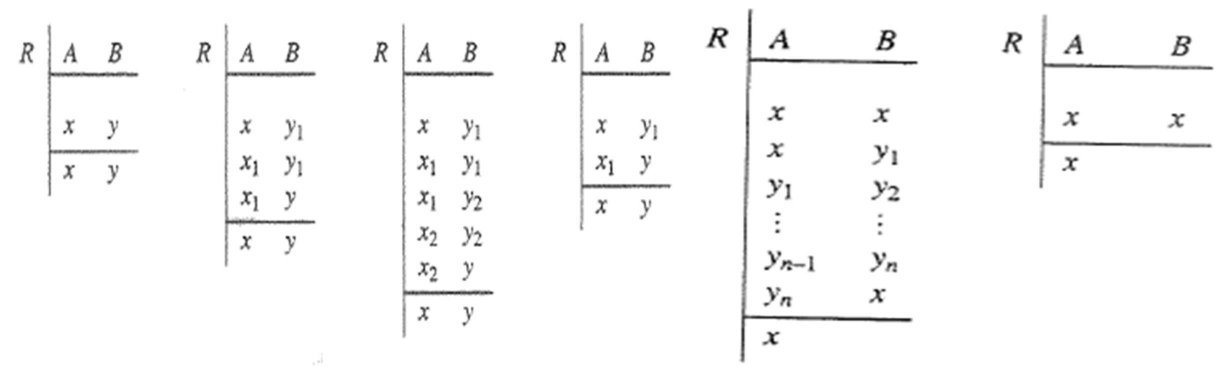
Teorie:

Definice: Nechť q1 = (T1, u1) a q2 = (T2, u2) jsou dva tablo dotazy. **Homomorfismus** z q2 na q1 je substituce θ taková, že θ(T2) ⊆ T1 a θ(u2) ⊆ u1.

Věta: q1 ⊆ q2 iff existuje homomorfismus z q2 na q1.

Definice: Řekneme, že tablo dotaz (T, u) je minimální, když neexistuje dotaz (S, v) ekvivalentní s (T, u) a |S|<|T| (ted ostře méně spojení).

Příklad z přednášky:



$q_0 = (T_0, \langle x, y\rangle)$ (a) $q_1 = (T_1, \langle x, y\rangle)$ (b) $q_2 = (T_2, \langle x, y\rangle)$ (c) $q_\omega = (T_\omega, \langle x, y\rangle)$ (d) $q = (T, u)$ (a) $q' = (T', u)$ (b)

⊆ ← ⊆ ← ⊆ ← ← = →

Substituce z (b) na (c):

Substituce z (c) na (d):

x1 v (d) je to samé jako x2 v (c).
x,y,y1 zůstává.
Je vidět, že (c) má v sobě všechny podmínky co má (d), takže výsledek bude podmnožinou výsledku (d)

Další příklady:

Bylo zadano, jak to tablo vypada

1. zjistit, zda existuje homomorfizmus mezi $q1=(T1,u1)$ a $q2=(T2,u2)$
2. najit $q2/R$ a $q1/R$ (nebo tak neco)

6. RDF

7. SPARQL

Dotazovací jazyky nad Webem.

<u>Příklad z přednášky:</u>

Consider the RDF dataset D:

D = { (B1, name, paul), (B1, phone, 777-3426),
(B2, name, john), (B2, email, john@acd.edu),
(B3, name, george), (B3, webPage, www.george.edu),
(B4, name, ringo), (B4, email, ringo@acd.edu),
(B4, webPage, www.starr.edu), (B4, phone, 888-4537), }

The following are graph pattern expressions and their evaluations over D according to the above semantics:

1. (1) P1 = ((?A, email, ?E) OPT (?A, webPage, ?W)).

$[[P1]]_D$ =

	?A	?E	?W
μ1	B2	john@acd.edu	
μ2	B4	ringo@acd.edu	www.starr.edu

2. P2 = (((?A, name, ?N) OPT (?A, email, ?E)) OPT (?A, webPage, ?W)).

$[[P2]]_D$ =

	?A	?N	?E	?W
μ1	B1	paul		
μ2	B2	john	john@acd.edu	
μ3	B3	george		www.george.edu
μ4	B4	ringo	ringo@acd.edu	www.starr.edu

3. P3 = ((?A, name, ?N) OPT ((?A, email, ?E) OPT (?A, webPage, ?W))).

$[[P3]]_D$ =

	?A	?N	?E	?W
μ1	B1	paul		
μ2	B2	john	john@acd.edu	
μ3	B3	george		
μ4	B4	ringo	ringo@acd.edu	www.starr.edu

Note the difference between $[[P2]]_D$ and $[[P3]]_D$. These two examples show that

$[[((A \text{ OPT } B) \text{ OPT } C)]]_D$!= $[[(A \text{ OPT } (B \text{ OPT } C))]]_D$ in general.

4. P4 = ((?A, name, ?N) AND ((?A, email, ?E) UNION (?A, webPage, ?W))).

Then

$[[P4]]_D$ =

	?A	?N	?E	?W

μ1	B2	john	john@acd.edu	
μ2	B3	george		www.george.edu
μ3	B4	ringo	ringo@acd.edu	
μ4	B4	ringo		www.starr.edu

5. P5 = (((?A, name, ?N) OPT (?A, phone, ?P)) FILTER ?P =777-3426).

Then
$[[P5]]_D$ =

	?A	?N	?P
μ1	B1	paul	777-3426

Další příklady:

- 6) Příklad na SPARQL viz. příklad 2) ve slajdech
- podrobněji vysvětlené a zavedené SPARQL, ze začátku není jasně vidět, že je to totéž, ale je
    - http://www.cambridgesemantics.com/semantic-university/sparql-by-example

Starší písemky

7.6.2007

Takže copak to tam dneska bylo: (prefix sou body)

1b - napsat nějakej pohled (vznosně to bylo nazváno virtuální relace)
1b - převést do RA (algoritmem!!!)
1b - převést do DRK (algoritmem!!!)

2b - napsat nějakej DRK (taky algoritmem!!!)

4b - Vysvětlit co znamená program v datalogu (typově nad spojema - metro)

1b - estli něco jako metro(linka, Můstek, Skalka) podle programu implikuje (linka, Skalka, Můstek) (neimplikovalo)
2b - kolik relací z programu de přepsat do RA

3b - napsat dotaz v rekutzivním SQL

3b - vypočítat dotaz ve SPARQL
?b- A nazávěr opět - napsat RA a DRK či co

dohromady za 19 bodů, pod 12 return;

13.6.2008

1. Máme schéma KNIHA(název, autor, isbn) a EXEMPLÁŘ(isbn, cena, země, ...)
Napište pohled v SQL DRAHEKNIHY(nazev, autor) - knihy s cenou na 4000 a zemí původu Velká Británie, Německo nebo Francie.
a) Převěďte pomocí algoritmu do RA (1 bod)
b) Převěďte a) do DRK pomocí algoritmu (1 bod)

2. Máme relace T(X,W), S(X,Y) a R(A,B) a dotaz v DRK
{y,w | ex. x (T(x,w) and (S(x,y) or R(x,y)))}
Prevedte pomoci algoritmu do RA (2 body)

3. DATALOG. Mame METRO(linka, stanice, nasledujici stanice)
A(x,x):-METRO(u,x,y)
A(x,y):-A(x,z),METRO(u,z,y)
B(x,z):-A(x,y),A(z,y),x!=z
O1(y):-A(y,Skalka),y!=Krizikova
O2(z):-B(z,Krizikova)

Co znamenaji A,B,O1 a O2? (4 body)

4. Ktery z A,B,O1 a O2 lze spocitat v RA? (2 body)

5. Mame tabulku Zamestnanci(jmeno, plat, vedouci). Najdete pomoci rekurzivniho dotazu vsechny zamestnance s platem nad 100 000, kteri jsou (i neprimi) podrizeni Ryby. (3 body)

6. R(A,B,C), S(C,D,E)
Dotaz v RA ((R*S)(B=2))[A,E] (2 body)

7. R|A|B|C   S|D|A|E   Q|D|F
----------   -----------   --------

|x|z|v |y|x|3 |y|2

U = <A:x,D:y,B:z>
a) Ekvivalentni dotaz v DATALOGU (1 bod)
b) Ekvivalentni dotaz v RA (1 bod)
c) Ekvivalentni dotaz v DRK (1 bod)

Celkem 19 bodu, 12 na projiti.

11.6.2009

Dnešní písemka byla obdobná výše uvedené s drobnými detaily...
1) Převod Algebra -> DRK - přirozené spojení, projekce, selekce
2) Převod DRK -> Algebra - obdoba
3) obdoba, navíc dotaz na stratifikaci a pár drobností
4) dtto
5) rekurzivní SQL - obdoba
6) Příklad na SPARQL viz. příklad 2) ve slajdech
Celkem 20 b, úspěch >=10.

8.6.2012 A

mame DB: Oddeleni(id_oddeleni, jmeno_oddeleni, id_vedouciho), zamestnanec(jmeno,id_zamestnance,jmeno_oddeleni)
- jmena oddeleni se mohla opakovat
- id vedouciho odkazuje do zamestancu
1. napsat view na : vyberte oddeleni, ktere se menuji stejne
2. rouzhodnout, zda je to view, co sme napsali v 1. promenlive (nebo modifikovatelne.. nebo tak neco.. sam sem byl zaskocen)
3. napsat dotaz 1. v DRK
4. Mame dotaz v DRK a rozhodnout, zda je/neni bezpecny
(neco jako) D(w,y)={Ex T(x,y) and (R(w,x) or S(x,y))} - odpoved: ano, protoze ma vsechny promenne omezene

5. napsat v DATALOGu (db. z 1. prikaldu) :
- to same co 1., tj: vybrat oddeleni s setejnymi jmeny
- vybrat zamestnance, kteri pracuji soucasne v oddeleni TV a oddeleni HUDBA
- vybrat zamestance, kteri pracuji bud v oddeleni OBUV nebo OBLEKY
- vybrat zamestnance, kteri nepracuji v oddeleni : TV, HUDBA, OBUV, OBLEKY
- udelat dotaz, ktery vypise vsech vedoucich (funkce mela byt tvaru NADRIZENY(ZAMESTANEC, VEDOUCI))
- vypsat top-vedoucich, kteri nemaji nikoho nadrazeneho (zrejmne se tam melo pouzit funkce NADRIZENY(ZAMESTANEC, VEDOUCI), pac to bylo za 0.5 bodu)
- jeste neco dalsiho... uz si nepamatuju

6. tablo dotazy
- napsat ekvivalentyi tablo dotaz k RA dotazu (R*S)(A,B)[C=4]
- napsat ekvivalentni tablo dotaz DRK dotazu: {x,y,z| R(x,2,z) and R(y,3,x) and R(z,y,1) }
- napsat ekvivalenti doraz k DATALOG dotazu : dotaz(x,y):- R(1,x,z),R(x,2,s)
Bylo zadano, jak to tablo vypada
- zjistit, zda existuje homomorfizmus mezi q1=(T1,u1) a q2=(T2,u2)
- najit q2/R a q1/R (nebo tak neco)

celkem 25 bodu, kto mel min nez 16, tak to nedal
Ty tabla byli za 9 bodu, takze sem mel smulu

Jen doplneni/oprava k predchozimu:

1. vybrat jmena oddeleni a jejich vedouci tech oddeleni, jejichz nazev se v databazi neopakuje
2. bylo to jestli je pohled aktualizovatelny, coz nebylo protoze v nem nebylo obsazeno ID ani zadna zavislost, ktera by ho urcovala

6a bylo lehce jinak, tusim ze (R*S)(B = 3)[A,E], pricemz relace vypadaly takto: R(A,B,C), S(C,D,E)

Jinak obsah pro me trochu prekvapivej, zadny XML, zadna rekurze, zadny RDF ani SPARQL, zadny produkcni operatory apod a hlavne zadna teorie, jen priklady... Ale pak tam bylo i nekolik veci, co se vykladalo uz v DJ I.

DOMÉNY
kvantifikátory umožňují svázat proměnnou s výskytem v nějaké
formuli
– formule∃xR(t1, t2, ..., x, ...) je vyhodnocena jako pravdivá, pokud
existuje doménové ohodnocení x takové, že n-tice (t1, t2, ..., x, ...) je prvkem R –
formule∀xR(t1, t2, ..., x, ...) je vyhodnocena jako pravdivá, pokud pro
všechna doménová ohodnocení x jsou n-tice (t1, t2, ..., x, ...) prvky R

 – např. dotaz {(film) | ∃nazev_kina KINO(nazev_kina, film) }vrátí názvy
všech filmůhraných v alespoňjednom kině
z důležité je určit, z jaké domény probíhá ohodnocování proměnných
při kvantifikaci
1. doména může být nespecifikovaná (tj. ohodnocení není omezeno
žádnou doménou) – ohodnocení typu universum
2. doména je typ příslušného atributu – ohodnocení podle domény
3. doména je množina hodnot daného atributu přítomných v relaci, ke které
se ohodnocení vztahuje – ohodnocení podleaktuální domény
Doménový kalkul
z např. dotaz {(film) | ∀nazev_kina KINO(nazev_kina, film) }se podle
způsobu ohodnocování proměnn énazev_kina(typu/domény string)může
lišit
– pokud ohodnocujeme podle univerza, dotaz nevrátí nic, protože v relaci
KINO určitě nebudou prvky nabývající všech možných hodnot v atributu
NAZEV_KINA (např. hodnoty 'kůň', 125, 'bflmpsvz' tam jistěnebudou)
– pokud ohodnocujeme podle domény, dotaz také pravděpodobněnevrátí
nic, protože v relaci KINO nebudou všechny hodnoty z domény string,
např. 'kůň', 'bflmpsvz', ...
– pokud ohodnocujeme podle aktuální domény, dotaz vrátí názvy všech
filmů, které se hrají ve všech kinech (přítomných v relaci KINO)

abychom předešli nekonečné kvantifikaci, je dobré zavést omezené kvantifikátory,
kteréomezují interpretaci vázaných proměnných
– místo ∃x (ϕ(x)) budeme používat ∃x (R(x) ∧ϕ(x))
– místo ∀x (ϕ(x)) budeme používat ∀x (R(x) ⇒ϕ(x))
z volné proměnnou v ϕ(x) lze rovněž omezit – konjunkcí
– R(x) ∧ϕ(x)
z pro bezpečné formule DRK platí:
1. dotaz neobsahuje ∀(není problém, ∀xϕ(x) lze nahradit ¬∃x(¬ϕ(x)))
2. pro disjunkciϕ1∨ϕ2
platí, že ϕ1 , ϕ2sdílí stejné volné proměnné
3. všechny volné proměnné v maximální konjunkci ϕ1∧ϕ2 ∧... ϕn

jsou omezené,tj. pro každou volnou proměnnou xplatí alespoňjedna z podmínek:
1. proměnná je volná v nějaké ϕi
, která není negací ani binárním porovnáním
(tj. ϕi je nenegovaná složená formule anebo nenegovaný „databázový“ predikát)
2. existuje $\phi i \equiv x = a$, kde a je konstanta
3. existuje $\phi i \equiv x = v$, kde v je omezená
4. negaci lze aplikovat pouze v konjunkcích bodu 3