Výpočet logických programov zdola nahor

Ján Šturc

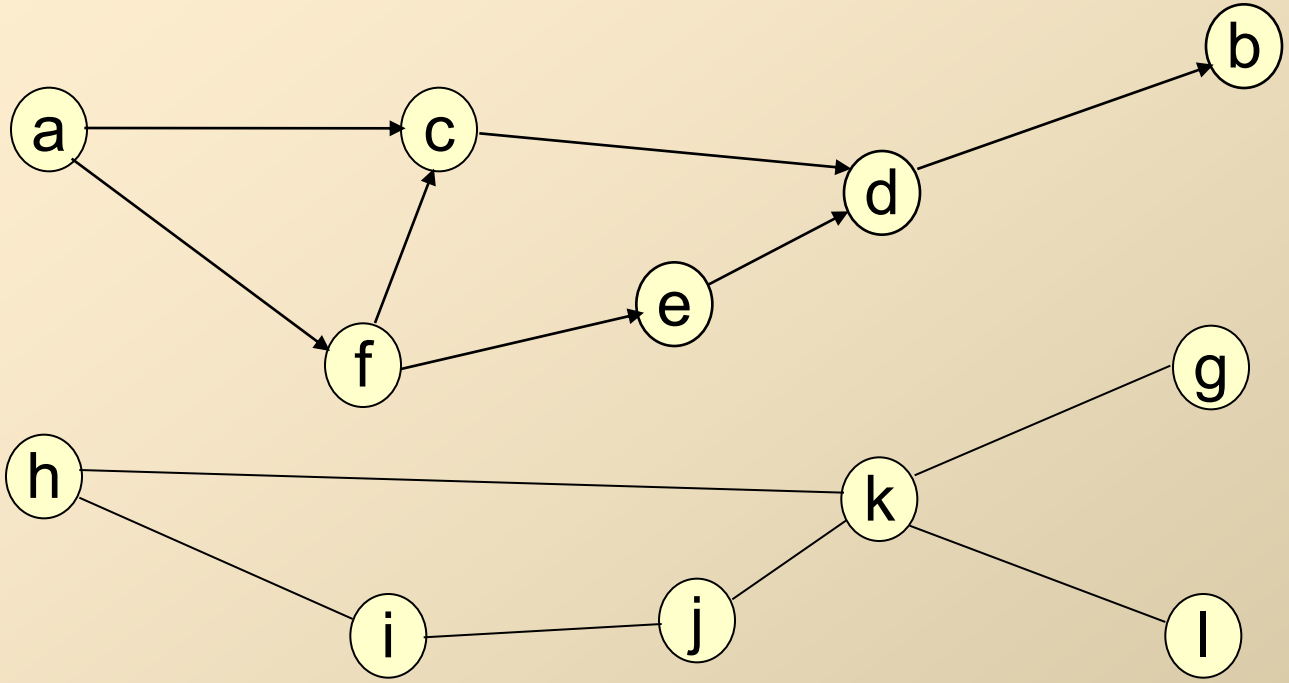
Datalóg a prológ

- Join Q(x,y,z) = R(x,y) ⋈ S(y,z)
  - q(x,y,z) ← r(x,y), s(y,z).
  - q(X,Y,Z):– r(X,Y), s(Y,Z),fail.
  - Datalóg počíta pomocou joinu. Prológ inštacijuje premenné pomocou SLD rezolúcie a tak postupne vypočítava n-tice joinu.
- Cesta v grafe

  p(x,y) ← e(x,y).
  p(x,y) ← e(x,z), p(z,y).
- Prológ vyžaduje program bez ľavej rekurzie. Datalóg s tým nemá problém. Na druhej strane uvažujme dotaz

  ?- p(a,b)

Príklad – inštancia grafu

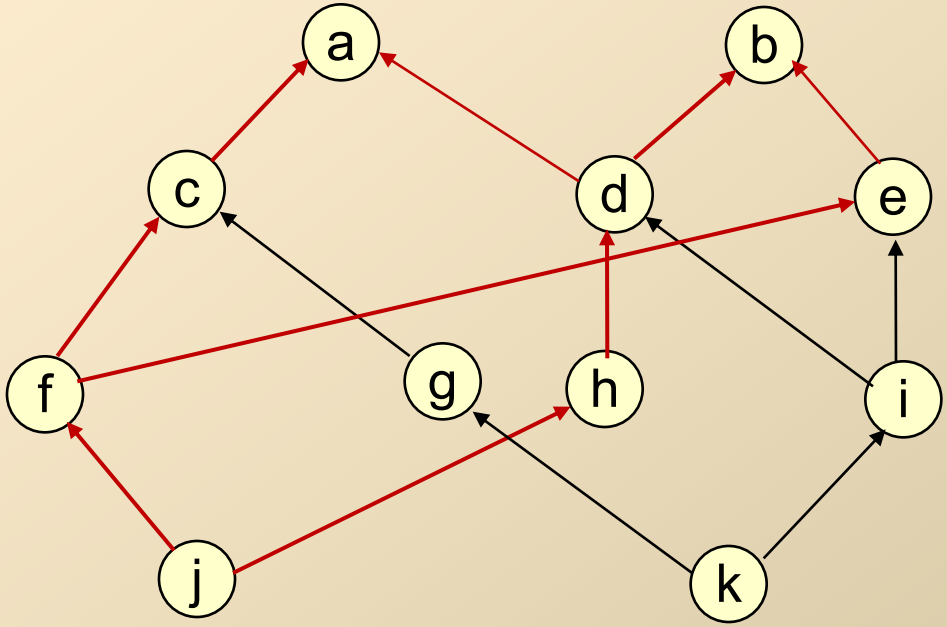


Orientácia:
1. z ľava do prava
2. obojsmerná (neorientovaný graf)

Iný príklad – predkovia Johna

r_1: anc(x,y) ← par(x,y) /* par(x,y): y je rodič x, anc – predok */
r_2: anc(x,y) ← par(x,z),anc(z,y)

Par:={ (c,a),(c,d),
(d,b),(e,b),
(f,c),(g,c),
(h,d), (i,d),
(f,e), (i,e),
(j,f), (k,g),
(j,h), (k,i) }

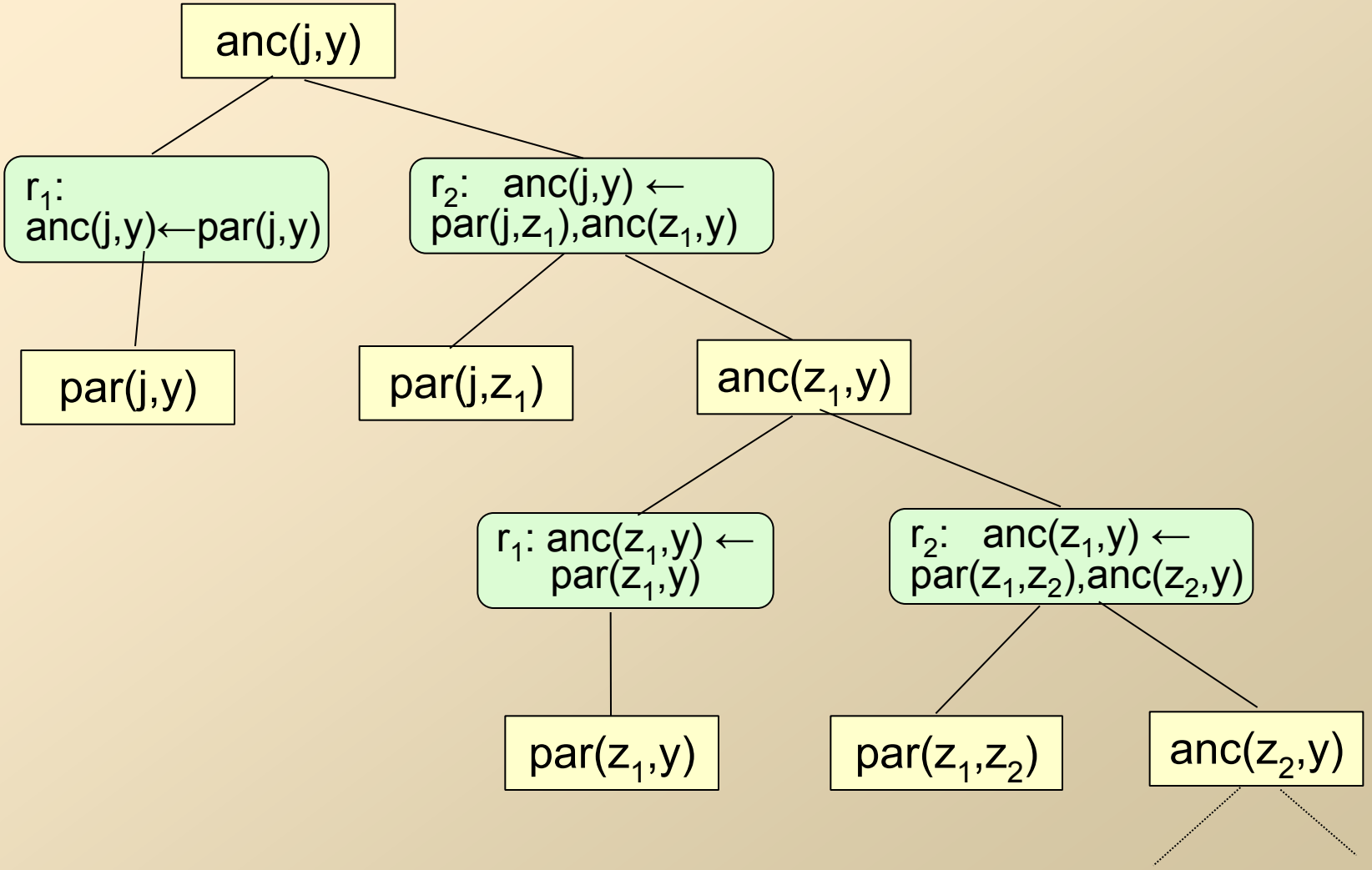


Môže existovať ľubovolne komplikovaný graf pod vrcholami f,g, h, i, j, k neovplyvní to výpočet zhora dolu. Zaťaží to ale metódu „buldozér“.

Strom pravidiel a cieľov (rule goal tree)

1. Koreň je cieľový uzol G_0.
2. Synovia cieľového uzla sú pravidlové uzly, pre pravidlá ktorých hlava sa unifikuje s daným cieľom. Musíme ošetriť premenné (podobne SLD rezolúcii).
   i. Pred unifikáciou premenné pravidla musia byť disjunktné s premennými cieľa.
   ii. Pri unifikácii preferujeme premenné cieľa.
   iii. Ak substituujeme za premennú v hlave, musíme tú istú substitúciu aplikovať aj na všetky výskyty premennej v tele pravidla.
   iv. Premenné, ktoré sa nevyskytujú v hlave sú lokálne v tele pravidla. (Budme ich indexovať.)
3. Synovia pravidlového uzla sú podciele jeho tela. Na pomenovanie premenných sa vzťahuje bod 2.iii.

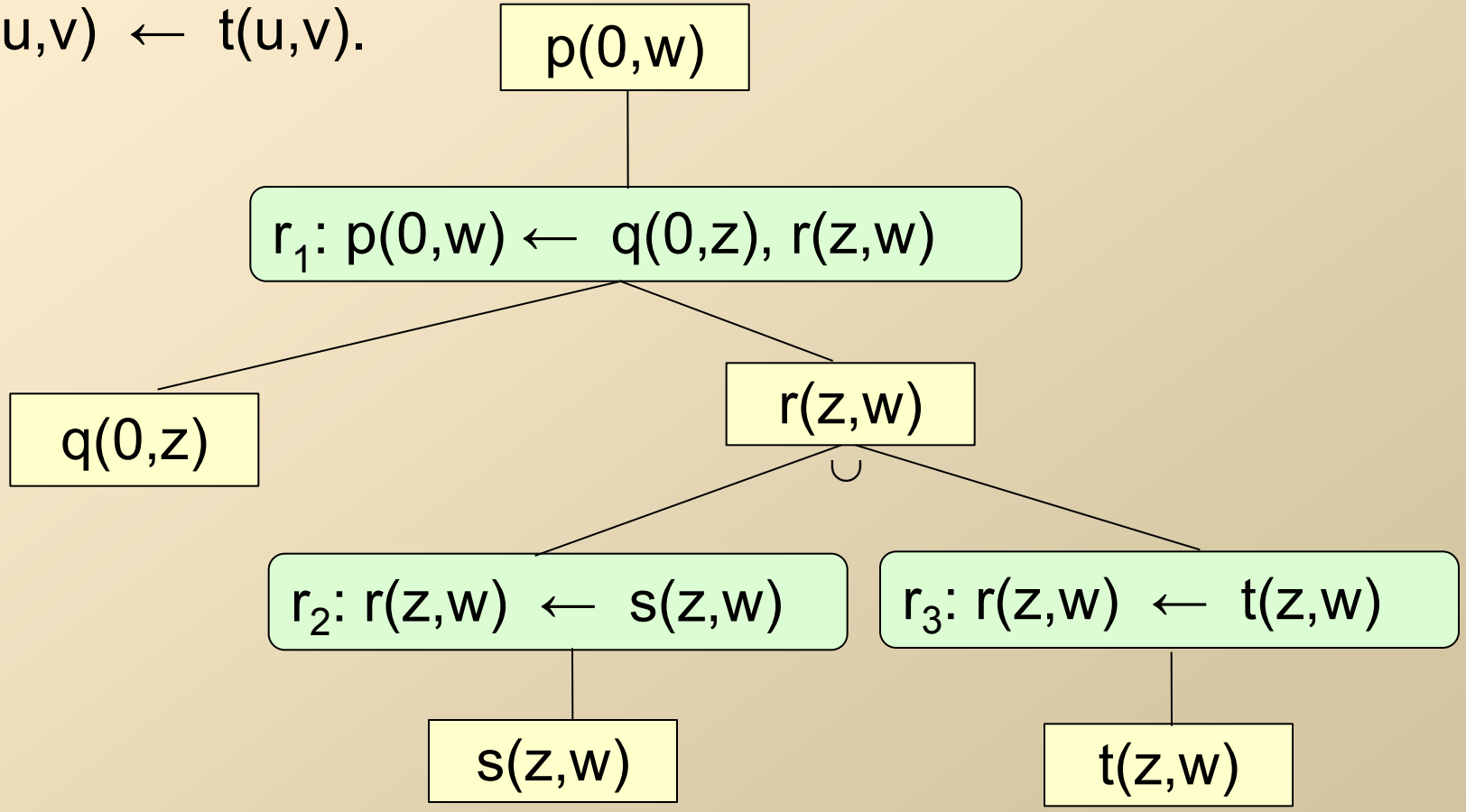
Príklad RGT pre predkov Johna



Nerekurzívny príklad

r_1: p(x,y) ← q(x,z), r(z,y).
r_2: r(u,v) ← s(u,v).
r_3: r(u,v) ← t(u,v).

Query: ?- p(0, w)



Simulácia výpočtu zhora nadol – šírenie väzieb a výsledkov

1. S každým cieľovým uzlom je spojená väzobná relácia M („*magic predicate*“). Obsahuje argumenty, ktoré sú v tomto uzle viazané. Nadobúdajú konštatné hodnoty, alebo konečný počet hodnôt ziskaných predošlým výpočtom (odovzdané stranou, sideway passed).
2. V každom pravidlovom uzle sú pomocné (*supplementary*) predikáty S_0, S_1, …. S_i je spojené s i-tým podcieľom tela. Reprezentujú tie viazané premenné, ktoré sa presúvajú z hlavy, respektíve z predošlých podcieľov.
3. Oba druhy uzlov obsahujú výsledkovú reláciu. Pre cieľový uzol je to množina tých n-tíc, ktoré zodpovedajú cieľu a dajú sa odvodiť na základe podstromu uzla. Pre pravidlový uzol, sú to hodnoty pre hlavu.

Simulácia výpočtu zhora nadol – vlastný výpočet

1. Vlastný výpočet sa sústreďuje do pravidlových uzlov.
2. Vo vnútorných cieľových uzloch a koreni sa robí len zjednotenie výsledkov prichádzajúcich z ich synov.
3. Listy predstavujú „selekciu – join“ väzobného predikátu s EDB reláciou.
4. V pravidlových uzloch sa ciele synov najprv konvertuju na relácie nad premennými pomocou operácie ATOV. Vypočíta sa ich „join / antijoin“. Výsledok pre hlavu sa konvertuje pomocou operácie VTOA. Takto získaný výsledok sa pošle otcovi (nadradenému cieľovému uzlu).

Výpočet súčasne s vytváraním stromu pravidiel a cieľov

- Vstup: Množina bezpečných pravidiel, EDB a dotaz – cieľ G_0.
- Výstup: Množina n-tíc (tuples), ktorá splňuje cieľ G_0 a vyplýva z danej množiny pravidiel a EDB.
- Metóda: Dve vzájomne rekurzívne procedúry.
  - *expandGoal(M, G, R)*, kde G je cieľ, M väzobný predikát pre niektoré argumenty G. R je výsledná relácia.
  - *expandRule(S_0, r, R)*, kde r je pravidlo, S_0 pomocný predikát, ktorý zaznamenáva niektoré viazané premenné pravidla r. R je opäť výsledná relácia.

procedure expandGoal(M, G, R)

{ if M = Ø **then** R:= Ø
else if G is a goal with an EDB predicate P **then** R:= M ⋈ P
else /* here G is a goal with an IDB predicate */
{ R:= Ø; /* in R the result has to be accumulated */
for each rule r whose head H unifies with G **do**
{ τ := mgu(G, H);
H' := $\prod_M H\tau$; /* τ(H) restricted to attributes of M */
S_0 := atov(H', M);
expandRule(S_0, τ(r), R_r);
R := R ∪ R_r
}
}
}

procedure expandRule(S_0, r, R)

{ let r is $H \leftarrow G_1, \ldots, G_k$;
for i:= 1 **to** k **do**
{ G_i' := $\Pi_{S_{i-1}}$ G_i; /* G_i restricted to arguments whose variables are attributes of S_{i-1} */
expandGoal(M_i, G_i, R_i);
Q_i := atov(G_i, R_i); /* convert R_i to variables */
S_i := $\Pi_T(S_{i-1} \bowtie Q_i)$; /* T is set of variables which appear in S_{i-1} or Q_i and in the rule r.*/
}
R := vtoa(H, S_k)
}

Vlastnosti algoritmu

- Rekurzívny algoritmu expandGoal-expandRule je presnou simuláciou algoritmu zhora nadol a zacyklí sa práve vtedy, keď sa zacyklí backtracking.
- Oprava: QRGT (queue based rule goal tree expansion). Robiť algoritmus po úrovniach (do šírky). Doplňovanie väzobných, pomocných relácii a výsledku o nové n-tice sa takto dá urobiť.
- QRGT konverguje k pevnému bodu pre bezpečné datalógové programy.
- QRGT pre cieľ G a väzbu M_0 vygeneruje n-ticu $\mu=p(t_1,\ldots,t_k)$ práve vtedy, ak
  - ju odvodí vyhodnocovanie zdola nahor pomocou tých istých pravidiel.
  - zodpovedá (match) cieľu G z viazanými atribútmi M_0.

Ozdoby pre predikáty a pravidlá

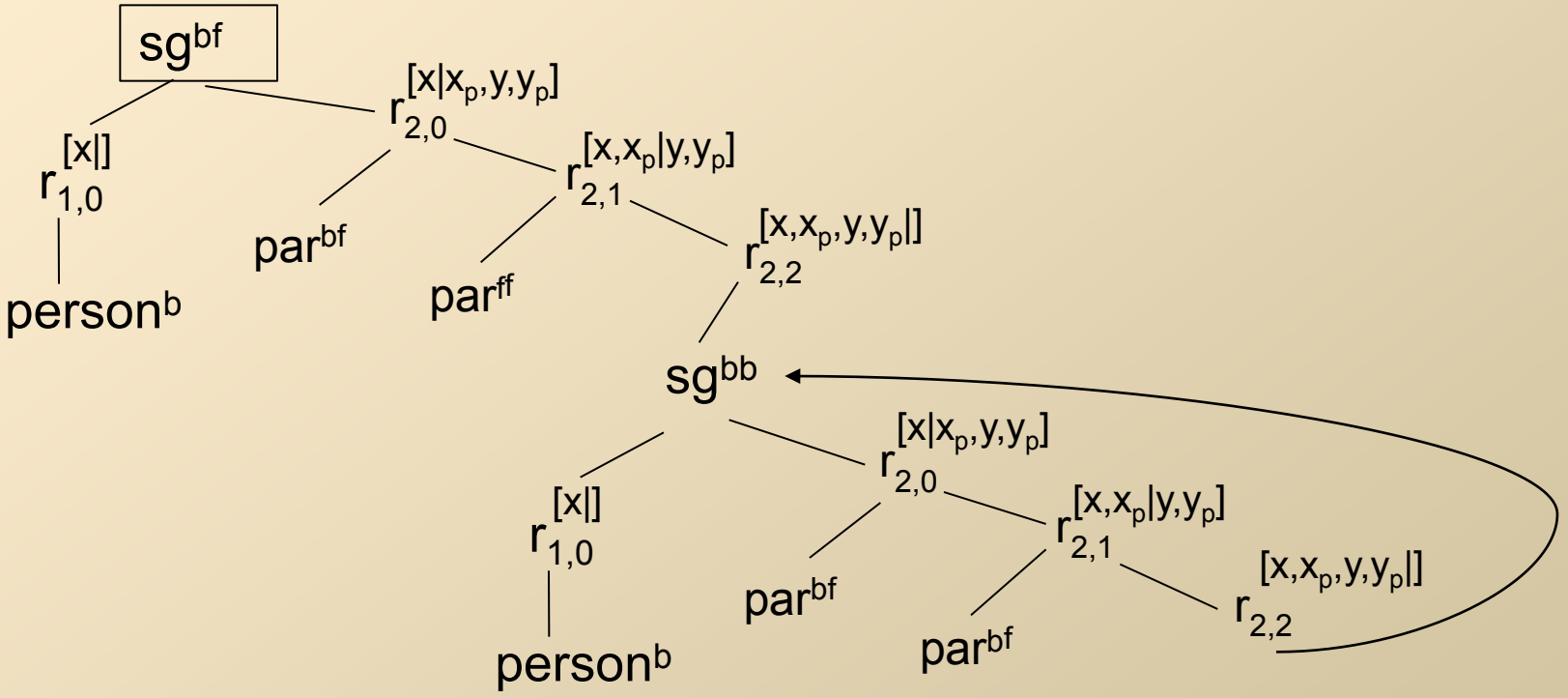
- Ozdobou pre predikát je slovo v abecede {b, f}. Ozdoba hovorí, ktoré argumenty obsahujú len viazané premenné b a ktoré sú volné f.
- Ozdoby využívame pri vyhodnocovaní pravidiel.
  - Premenná, korá sa vyskytuje vo viazanom argumente hlavy je ohraničená (nadobúda konečne mnoho hodnôt) pred vyhodnotením ľubovolného podcieľa tela.
  - Premenná x je viazaná po vyhodnotení podcieľa G_i, ak premenná x bola viazaná pred vyhodnotením podcieľa G_i, alebo sa vyskytuje kdekoľvek v argumentoch G_i.
- Ozdoba pre pravidlo je: [<zoznam viazaných premenných>|<zoznam volných premenných].

Graf cieľov a pravidiel – rule goal graph (RGG)

1. Cieľový uzol s EDB predikátom nemá výstupné hrany.
2. Z cieľového uzla s ozdobeným IDB predikátom p^{α} vedú hrany do pravidlových uzlov takých, že ich hlava je predikát p a majú viazané premenné, ktoré sa vyskytujú v argumentoch viazaných ozdobou α. $r_0^{[x_1, \ldots, x_m | y_1, \ldots, y_n]}$
3. Z pravidlového uzla $r_i^{[x_1, \ldots, x_m | y_1, \ldots, y_n]}$, $i \geq 0$, vedie hrana do
   a) cieľového uzla q^{β}, kde q je i-tý predikát tela pravidla a β je ozdoba, ktorá viaže argumenty, ktoré obsahujú len premenné, ktoré sú viazané v tele pravidla.
   b) ak i nie je posledný predikát, potom hrana do pravidlového uzla r_{i+1}, ktorý má navyše viazané všetky premenné vyskytujúce sa v argumentoch q.

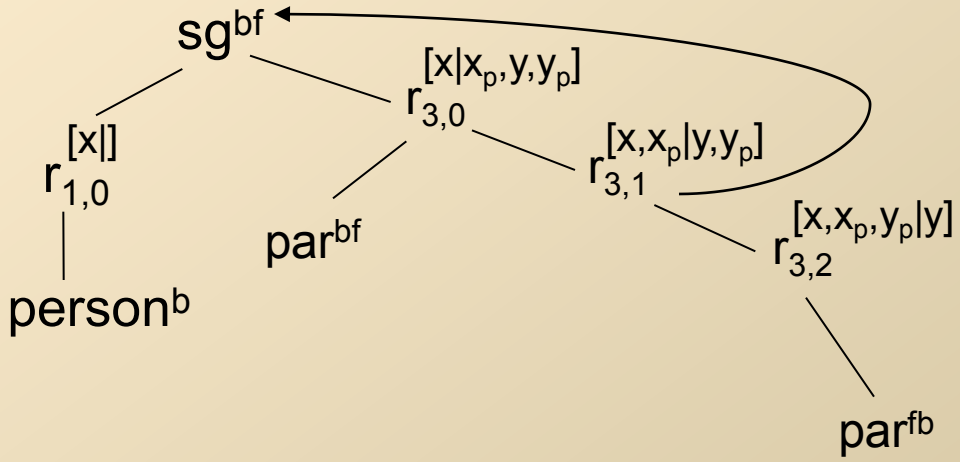
Príklad – tá istá generácia (same generation)

r_1: sg(x,x) ← person(x)
r_2: sg(x,y) ← par(x,x_p), par(y,y_p), sg(x_p,y_p)



Vylepšenie preusporiadaním podcieľov v pravidle

r_1: sg(x,x) ← person(x)
r_3: sg(x,y) ← par(x,x_p),sg(x_p,y_p),par(y,y_p)



Ako ďalej? Úpravy, ktoré pomáhajú

- Uniformita ozdôb = rovnaké väzby v pravidlách (making binding patterns unique)
- Rektifikácia podcieľov. Rektifikácia hláv pravidiel, ktorú robíme kvôli výpočtu pevného bodu nestačí. Problém je viacnásobný výskyt tej istej premennej v podcieli (*rectifying subgoals*).
- Preusporiadanie podcieľov v pravidlách (*reordering subgoals*).
- Nie každý graf cieľov a pravidiel sa dá výpočtove realizovať (dovolené poradia ozdobených predikátov). Potrebujeme test realizovateľnosti (*the feasibility problem for rule-goal graph*).

Transformácia na program s uniformnými ozdobami

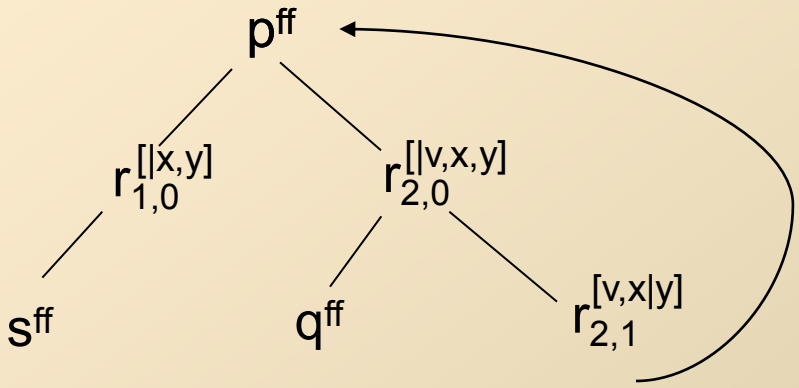
1. Pre každý ozdobený predikát p^{α}, vyskytujúci sa v RGG, vytvor nový predikát p_α.
2. Pre každé pravidlo s hlavou p, vytvor kópiu tohto pravidla r_α s hlavou p_α.
3. Vo všetkých uzloch daného pravidla r_0, r_1, …, r_k uprav synov (cieľové uzly)
   a) Ak predikát q v uzle q^{β} je EDB predikát alebo zabudovaný predikát, ponechaj tento predikát v r_α tak, ako bol v r.
   b) Ak predikát q v uzle q^{β} je IDB predikát, zmeň ho na q_β.

Príklad – tá samá generácia (slide 16)

r_1_bf: sg_bf(x,x) ← person(x)
r_2_bf: sg_bf(x,y) ← par(x,x_p),par(y,y_p),sg_bb(x_p,y_p)
r_1_bb: sg_bb(x,x) ← person(x)
r_2_bb: sg_bb(x,y) ← par(x,x_p),par(y,y_p),sg_bb(x_p,y_p)

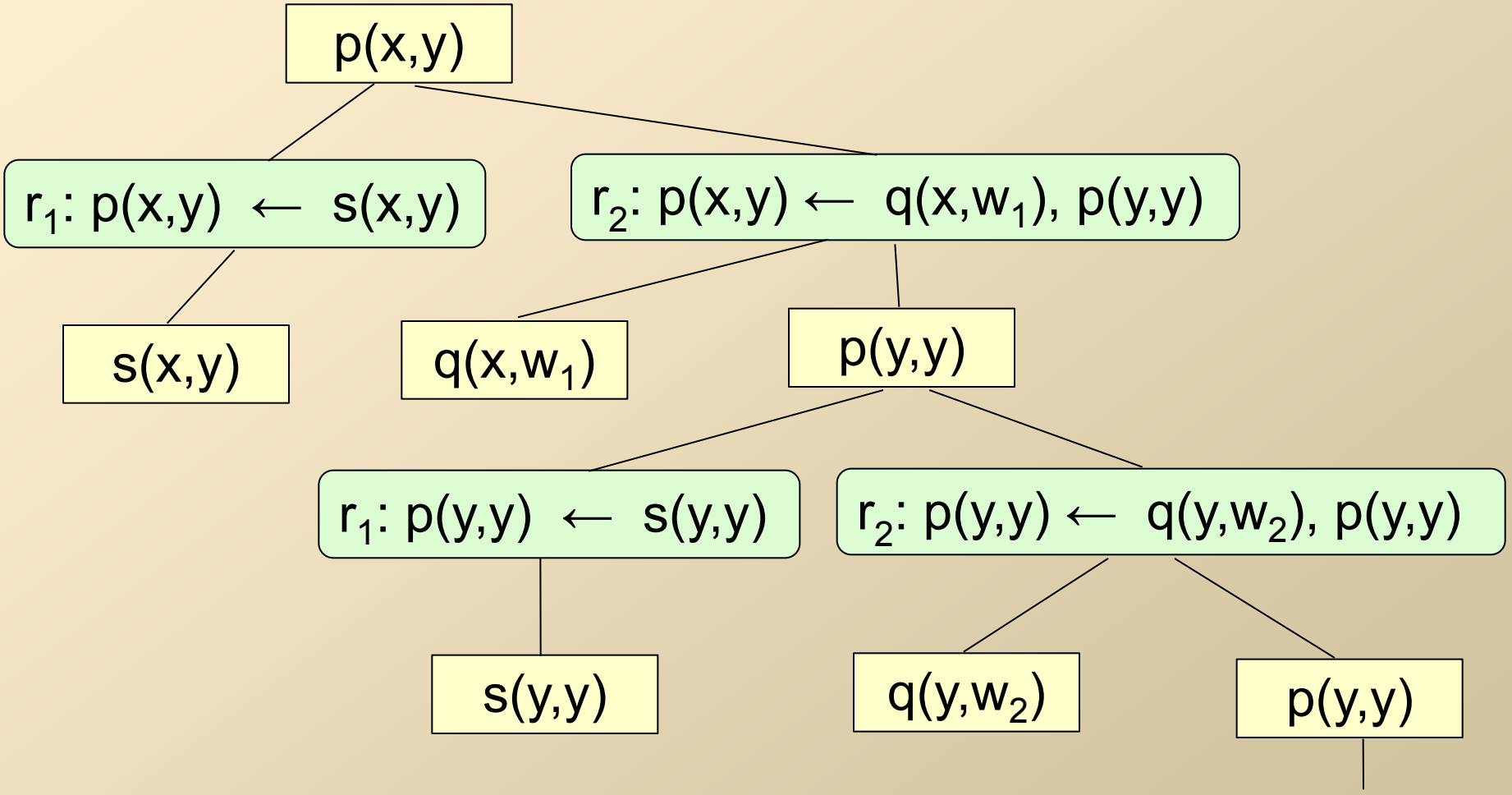
Iný príklad

r_1: p(x,y) ← s(x,y)
r_2: p(x,y) ← q(x,v), p(y,y)



V tomto grafe je pravidle $r_{2,1}$ nesprávna ozdoba predikátu mala by byť p^{bb}. Spôsobilo to dvojnásobné použitie tej istej premennej y.

Strom cieľov a pravidiel pre predošlý príklad



Príklad pokračovanie pravidlá s rektifikovanými podcieľmi

r_1: $p(x,y) \leftarrow s(x,y)$.
r_2: $p(x,y) \leftarrow q(x,v), p_1(y)$.
r_3: $p_1(y) \leftarrow s(y,y)$.
r_4: $p_1(y) \leftarrow q(y,v), p_1(y)$.

Pre tieto pravidlá sa už dá urobiť „korektný“ graf cieľov a pravidiel. Bude pozostávať z troch podobných „poschodí“. Pre p^{ff}, p_1^{f} a p_1^{b}. Až „posledné poschodie“ sa zacyklí.

Algoritmus rektifikácie podcieľov

while existuje podcieľ g s opakovanými argumentami alebo konštantami **do {**

1. Nech g je podcieľ s predikátom p a opakovanými argumentami alebo konštantami vytvor nový predikát p', odstránením opakovaných argumentov a konštánt.
2. Pre každé pravidlo r s hlavou p vytvor nové pravidlo tak, že vypočítaš τ = mgu(g, p), pričom sa uprednostňujú premenné podcieľa. Nové pravidlo dostaneme z rτ tak, že všetky predikátu p s argumentami g nahradíme predikátom p'.
3. Nahraď v pôvodných pravidlách podcieľ g podcielom p' **}**

Tento proces musí skončiť. V každej iterácii sa zmenší počet argumentov v nejakom podcieli. Navyše ak pôvodné pravidlá boli bezpečné aj výsledok je bezpečný.

Algoritmus: Zuniformnenie ozdôb – making binding patterns unique

Vstup: Množina pravidiel a cieľ (dotaz).
Výstup: Modifikovaná množina pravidiel zabezpečujúca, že v strome cieľov a pravidiel má každý výskyt IDB predikátu rovnaké väzby. (V RGG tie isté ozdoby).

Metóda:

1. Rektifikuj podciele v pravidlách
2. Transformuj na program s uniformnými ozdobami (slide 19)

Usporiadanie podcieľov

1. Zabudované predikáty sa obvykle nedajú „vyhodnotiť“ pokiaľ argumenty „vstupnej množiny“ nie sú viazané.
2. Niekedy môžeme aj pre EDB predikáty preferovať takýto prístup napr. z dôvodu indexu, alebo keď EDB predikát nie je definovaný tabuľkou ale programom.
3. V prípade IDB predikátov definovaných pomocou funkčných symbolov napr. nat(x). Nedokážeme nat^f.
4. Negované predikáty. Všetky argumenty musia byť viazané. (Túto podmienku môžeme trochu oslabiť, len na premenné, ktoré sa vyskytujú aj inde v pravidle. Aj tak pravidlo $p(x) \leftarrow q(x), \neg r(y)$ je problém. Prelož do SQL.)

Dôsledkom je, že predikáty v tele pravidiel a RGG nemôžu byť v ľubovolnom poradí. Niektoré poradia sú „lepšie“ ako iné.

Predpoklad viazané je lepšie – the bound is easier assumption

Definujeme čiastočné usporiadanie ozdôb: $\alpha \leq \beta$, ak β obsahuje b aspoň na tých miestach, kde ich má α.

Predpokladáme: ak $\alpha \leq \beta$ a vieme vyhodnotiť p^{α}, potom vieme vyhodnotiť aj p^{β} a vyhotenie p^{β} nebude výpočtove náročnejšie ako p^{α}.

Ozdobu α pre, ktorú vieme predikát vyhodnotiť nazývame dovolenou.
Dôsledky uvedených definícii:
1. Pre každý predikát existuje množina mininimálnych dovolených ozdôb.
2. Poradie predikátov môžeme vyberať pažravou (greedy) metódou.

Usporiadanie podcieľov v pravidle

Nech je dané:

- Pravidlo $H \leftarrow G_1, \ldots, G_m$
- Ozdoba pre hlavu H
- Dovolené ozdoby pre predikáty tela

Chceme nájsť poradie ozdobených predikátov v tele, že je:

- dovolené, t.j. každý predikát má dovolenú ozdobu, a
- korektné, t.j. aspoň argumenty predpísané ozdobou sú v danom mieste výpočtu viazané.

Algoritmus **A**:

Udržujeme si množinu viazných premenných a v každom bode výpočtu testujeme, či existuje predikát, ktorý má dovolenú ozdobu z danej množiny. Ak neexistuje, pravidlo s danou hlavou sa nedá realizovať.

Bez ďalších ohraničení je to backtracking, ale za predpokladu „bound is easier“ sa nikdy nemusíme vracať.

Algoritmus na testovanie realizovateľnosti RGG

Vstup: Množina pravidiel. Cieľ (dotaz) p^{α}, zoznam dovolených ozdôb pre EDB predikáty.

Výstup: RGG pre p^{α}, ak existuje, inak správa, že nexistuje.

Metóda: Budeme používať algoritmus A a predpoklad „bound is easier“. Zavedieme dve množiny: Množinu F zakázaných ozdobených cieľov. Na počiatku obsahuje všetky EDB predikáty s ozdobami, ktoré nie sú dovolené. Množinu I zaujímavých IDB cieľov. Na počiatku obsahuje len p^{α}. Následne použijeme algoritmus A na nejaký cieľ v množine I. Ak sa nám nepodarí nájsť realizovateľné usporiadanie tela niektorého pravidla presunieme tento cieľ do množiny F. Ak algoritmus A pre nejaké pravidlo uspeje, vložíme do množiny I požadované IDB ciele.

Základný cyklus algoritmu

for each adorned goal $q^{\beta} \in$ I **do**
  for each rule r with the head q **do**
{ **A**(r, q^{β}, F); /* poradie podcieľov v r */
  if an ordering found **then**
    add to I adorned IDB predicates in selected order
  else move q^{β} from I to F;
}

Za predpokladu „bound is easier“ môžeme F reprezentovať maximálnymi prvkami t.j. predpokladať, že s každým prvkom obsahuje všetky menšie. Všetky väčšie prvky sú dovolené.

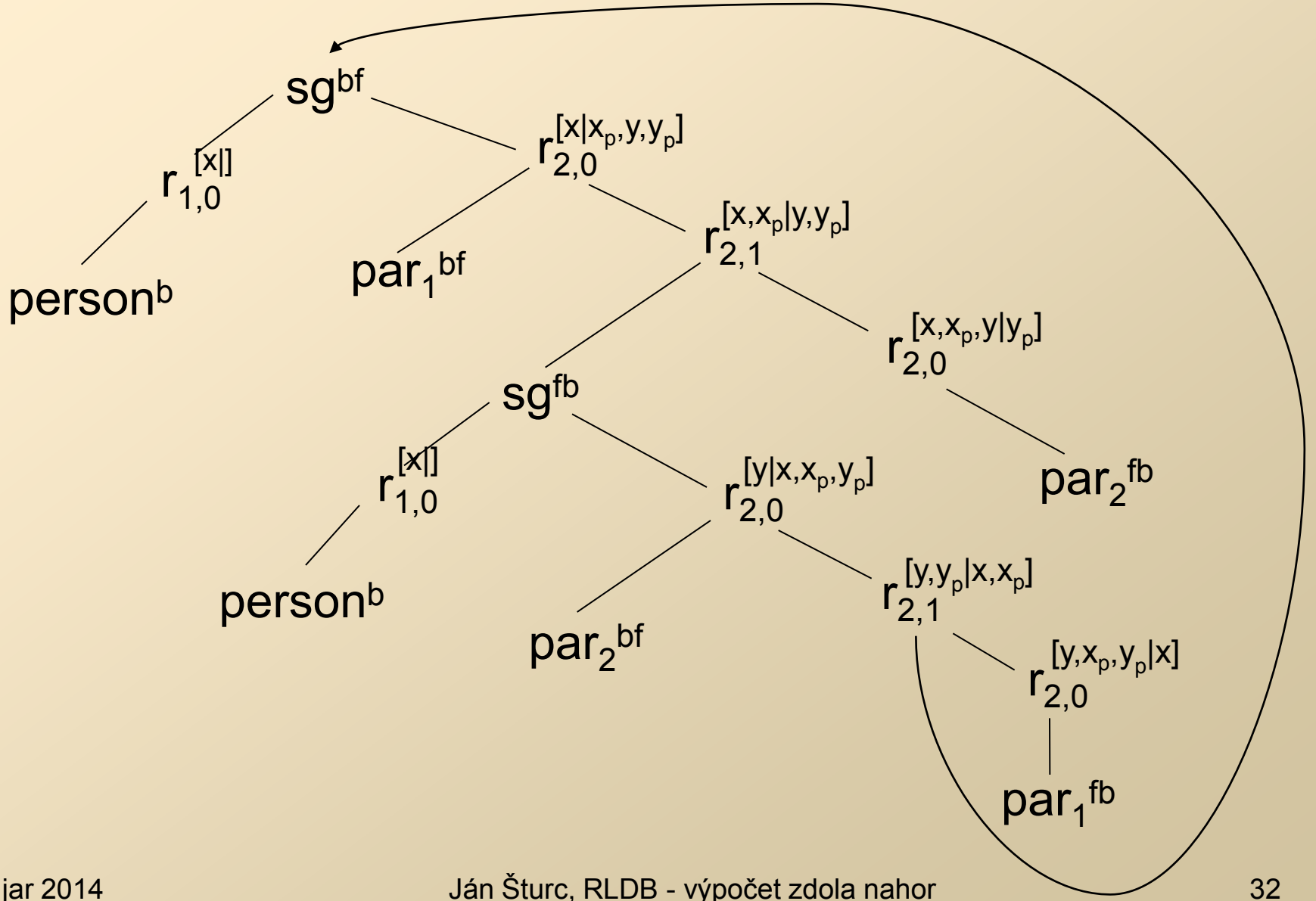
Algoritmus skončí keď
1.presunieme p^{α} do F (neúspech, neexistuje feasible RGG)
2.základný cyklus nezmení I (všetky pravidlá sa dajú realizovať s ozdobami v I.)

Príklad – tá istá generácia

r_1: $sg(x,x) \leftarrow person(x)$
r_2: $sg(x,y) \leftarrow par(x,x_p), par(y,y_p), sg(y_p,x_p)$
cieľ: sg(john, w)

I = $\{sg^{bf}\}$; F = $\{par^{ff}, person^{f}\}$
Pre r_1 je jediné možné usporiadanie s ozdobou b $person^{b}$.
Pre r_2 vyberie poradie tela par_1^{bf}, sg^{fb}, par_2^{fb}. V dôsledku pridá sg^{fb} do I. V nasledujúcom kole r_1 dopadne rovnako a r_2 pre sg^{fb} bude par_2^{bf}, sg^{bf}, par_1^{fb}. Žiadne nové ozdobené ciele sa nepridajú. I = $\{sg^{bf}, sg^{fb}\}$.

Rule goal graph pre príklad



Najkratšia cesta (negácia)

r_1: p(x,y,d) ← e(x,y,d).
r_2: p(x,y,d) ← sp(x,z,d_1),e(z,y,d_2), d=d_1+d_2.
r_3: sp(x,y,d) ← p(x,y,d),¬p(x,y,c), c<d.
?- sp(a,v,d)

Optimalizácia pravidla r_3:
r_4: sp(x,y,d) ← p(x,y,d),¬p(x,y,c≥d).
r_5: sp(x,y,d) ← subtotal(p(x,y,d); x; d=min(d)), p(x,y,d).

Graf cieľov a pravidiel