Některé teoretické pojmy logického programování

Upraveno podle *Lloyd, J. W.: Foundations of logic programming. Berlin, Springer-Verlag 1987.*

- Termy, formule a klauzule (→ Logika).
- Substituce a unifikace, unifikační algoritmus.
- Princip vyvracení, rezoluční princip (→ Logika), SLD rezoluce.

Termy, formule a klauzule (I)

Každá abeceda 1.řádu se skládá ze sedmi tříd symbolů:

- proměnné
- predikátové symboly
- funkční symboly
- konstanty (funkční symboly arity 0)
- spojky
- kvantifikátory
- pomocné symboly

Termy, formule a klauzule (II)

Term je jedno z:

- konstanta
- proměnná
- funkční symbol s argumenty (funkční struktura).

Predikátový symbol s argumenty je **atomická formule** (atom).

Literál je buď atomická formule (pozitivní literál), nebo její negace (negativní literál).

$$A \qquad \neg A \quad non\, A \quad \bar{A}$$

Termy, formule a klauzule (III)

Klauzule je formule ve tvaru

$$\forall X_1 \forall X_2 \ldots \forall X_s \quad (L_1 \vee L_2 \vee \ldots \vee L_m)$$

kde každé L_i je literál, a $X_1, X_2, \ldots, X_s$ jsou všechny proměnné v $L_1 \vee L_2 \vee \ldots \vee L_m$ se vyskytující.

Klauzule je i (kvantifikovaný) literál (m=1) a prázdná klauzule (m=0).

Klauzuli

$$\forall X_1 \ldots \forall X_s \quad (A_1 \vee \ldots \vee A_k \vee \neg B_1 \vee \ldots \vee \neg B_n)$$

kde všechny A_i, B_j jsou atomy, a $X_1, \ldots, X_s$ jsou všechny proměnné v těchto atomech se vyskytující, budeme zapisovat jako

$$A_1, \ldots, A_k \leftarrow B_1, \ldots, B_n.$$

Termy, formule a klauzule (IV)

Příklad klauzule (Prolog):

$\mathrm{bratr(X,Y)}$:- $\mathrm{muz(X)}$,$\mathrm{otec(Z,X)}$,$\mathrm{otec(Z,Y)}$.

$\forall$ x,y ((M(x) $\wedge$ ($\exists$ z O(z,x) $\wedge$ O(z,y))) $\Rightarrow$ B(x,y))

$\forall$ x,y $\exists$ z ((M(x) $\wedge$ O(z,x) $\wedge$ O(z,y)) $\Rightarrow$ B(x,y))

$\forall$ x,y $\exists$ z ($\neg$ (M(x) $\wedge$ O(z,x) $\wedge$ O(z,y)) $\vee$ B(x,y))

$\forall$ x,y $\exists$ z $\neg$ M(x) $\vee$ $\neg$ O(z,x) $\vee$ $\neg$ O(z,y) $\vee$ B(x,y)

$\forall$ x,y $\neg$ M(x) $\vee$ $\neg$ O(f(x,y),x) $\vee$ $\neg$ O(f(x,y),y) $\vee$ B(x,y)

Termy, formule a klauzule (V)

Hornova klauzule obsahuje nejvýše jeden pozitivní literál.

Programová (definitní) klauzule obsahuje právě jeden pozitivní literál:

$$A \leftarrow B_1, \ldots, B_n.$$

Cílová klauzule (cíl) obsahuje pouze negativní literály:

$$(false) \leftarrow B_1, \ldots, B_n.$$

Jednotková klauzule (fakt) obsahuje pouze pozitivní literál:

$$A \leftarrow (true).$$

Prázdná klauzule (spor - $\square$) :

$$(false) \leftarrow (true).$$

Substituce a unifikace (I)

Substituce θ je konečná množina tvaru

$$\{u_1/s_1, \ldots, u_m/s_m\}$$

kde každé u_i je proměnná, každé s_i je term různý od u_i, a proměnné $u_1, \ldots, u_m$ jsou navzájem různé
($\rightarrow$ zobrazení z množiny proměnných do množiny termů!).

Výraz je term, literál, nebo konjunkce/disjunkce literálů; **jednoduchý výraz** je term nebo atom.

Nechť E je výraz a $\theta = \{u_1/s_1, \ldots, u_m/s_m\}$ je substituce. Potom $E\theta$, **instance** výrazu E pomocí θ, je výraz, který vznikne z E simultánním nahrazením všech výskytů každé proměnné u_i v E termem s_i.

Příklad: $E = p(X, Y, f(a))$, $\theta = \{X/b, Y/X\}$. Potom $E\theta = p(b, X, f(a))$.

Substituce a unifikace (II)

Nechť $\theta = \{u_1/s_1, \ldots, u_m/s_m\}$
a $\sigma = \{v_1/t_1, \ldots, v_n/t_n\}$ jsou substituce. Potom **složení** $\theta\sigma$ substitucí θ a σ je substituce, která vznikne z množiny

$$\{u_1/s_1\sigma, \ldots, u_m/s_m\sigma, v_1/t_1, \ldots, v_n/t_n\}$$

vynecháním vazeb $u_i/s_i\sigma$, pro něž $u_i = s_i\sigma$
($\rightarrow$ dosazení z σ "vrací zpět" dosazení z θ);
a vazeb v_j/t_j, pro něž $v_j \in \{u_1, \ldots, u_m\}$
($\rightarrow$ dosazení z θ odstraňuje proměnnou, za kterou má dosazovat σ).

Příklad: $\theta = \{X/f(Y), Y/Z\}$, $\sigma = \{X/a, Y/b, Z/Y\}$.
Potom $\theta\sigma = \{X/f(b), Z/Y\}$.

Substituce ϵ určená prázdnou množinou se nazývá **identická substituce**.

Substituce a unifikace (III)

Unifikace (Herbrand 1930):

Nechť S je konečná množina jednoduchých výrazů. Substituce θ se nazývá **unifikátor** pro S, jestliže $S\theta$ je jednoprvková množina.

θ je **nejobecnější unifikátor** pro S, $nu(S)$, jestliže ke každé σ, která je unifikátorem pro S, existuje substituce γ taková, že $\sigma = \theta\gamma$.

Příklad: Pro $S = \{p(f(X), Z), p(Y, a)\}$ je unifikátorem např. $\{Y/f(a), X/a, Z/a\}$;
$nu(S) = \{Y/f(X), Z/a\}$.

Pro každé S je $nu(S)$ určen *jednoznačně*, až na přejmenování proměnných.

Substituce a unifikace (IV)

Množina neshod v konečné množině jednoduchých výrazů S:

Nalezněte první pozici zleva, v níž existuje neshoda, a vyčleňte z každého výrazu v S podvýraz začínající symbolem na této pozici. Množina všech takových podvýrazů je množina neshod.

Příklad: Pro $S = \{p(f(X), h(Y), a),$
$p(f(X), Z, a),$
$p(f(X), h(Y), b)\}$
je množinou neshod $\{h(Y), Z\}$.

Substituce a unifikace (V)

Unifikační algoritmus pro konečnou množinu jednoduchých výrazů S:

1. Polož $k = 0$, $\sigma_0 = \epsilon$.
2. Jestliže je $S\sigma_k$ jednoprvková množina, pak STOP; σ_k je $nu(S)$.
   Jinak nechť D_k je množina neshod v $S\sigma_k$.
3. Jestliže D_k obsahuje taková v a t, že v je proměnná *nevyskytující se* v t, polož $\sigma_{k+1} = \sigma_k\{v/t\}$, zvětši k o 1, a jdi na 2.
   Jinak STOP; S není unifikovatelná.

Unifikační teorém: Jestliže S je unifikovatelná, pak se unifikační algoritmus zastaví a vydá $nu(S)$. Jestliže S není unifikovatelná, pak se rovněž zastaví a vydá o tom zprávu.

Substituce a unifikace (VI)

Příklad: Nechť $S = \{p(f(a), g(X)), p(Y, Y)\}$.

0. $\sigma_0 = \epsilon$.
1. $D_0 = \{f(a), Y\}$, $\sigma_1 = \{Y/f(a)\}$,
   $S\sigma_1 = \{p(f(a), g(X)), p(f(a), f(a))\}$.
2. $D_1 = \{g(X), f(a)\}$.
   Tudíž S není unifikovatelná.

Příklad: Nechť $S = \{p(a, X), p(Y, h(Y))\}$.

0. $\sigma_0 = \epsilon$.
1. $D_0 = \{a, Y\}$, $\sigma_1 = \{Y/a\}$,
   $S\sigma_1 = \{p(a, X), p(a, h(a))\}$.
2. $D_1 = \{X, h(a)\}$, $\sigma_2 = \{Y/a, X/h(a)\}$,
   $S\sigma_2 = \{p(a, h(a)), p(a, h(a))\} = \{p(a, h(a))\}$.
   Tudíž S je unifikovatelná a σ_2 je $nu(S)$.

Substituce a unifikace (VII)

Příklad: Nechť $S = \{p(X, X), p(Y, f(Y))\}$.

0. $\sigma_0 = \epsilon$.
1. $D_0 = \{X, Y\}$, $\sigma_1 = \{X/Y\}$,
   $S\sigma_1 = \{p(Y, Y), p(Y, f(Y))\}$.
2. $D_1 = \{Y, f(Y)\}$.
   Protože se Y vyskytuje v $f(Y)$, S není unifikovatelná!

Kontrola výskytu ("occur-check") v unifikačním algoritmu ovšem může (v nejhorším případě) vést na exponenciální složitost vzhledem k délce vstupu!

→ v implementacích Prologu se většinou vypouští
→ možnost "unifikačního" nekonečného cyklu!
Např.:

?- X = f(X).

Rezoluce a vyvracení (I)

Rezoluční princip: odvozovací pravidlo se schématem

$$\frac{X \vee Z, \;\; Y \vee \neg Z}{X \vee Y}$$

V logickém programování je Z zpravidla atom; $\{Z, \neg Z\}$ se nazývá **doplňkový pár** literálů; klauzule $X \vee Z$ a $Y \vee \neg Z$ se nazývají **rodičovský pár**, a $X \vee Y$ je jejich **rezolventa**.
Rezoluce je **korektní** - rezolventou dvou pravdivých klauzulí je opět pravdivá klauzule.

Princip vyvracení: Uzavřená formule ϕ vyplývá z teorie T právě tehdy když $T \cup \{\neg\phi\}$ je nesplnitelná (kontradiktorická).

Rezoluce a vyvracení (II)

Příklad: Chceme dokázat, že formule u vyplývá z teorie

$$T = \{v \Rightarrow u, \; w \Rightarrow v, \; w\}$$

Vyvracení: Převedeme T na klauzulární tvar, a snažíme se dokázat spornost teorie

$$T' = \{\neg v \vee u, \; \neg w \vee v, \; w \vee \mathbf{false}, \; \neg u \vee \mathbf{false}\}$$

(ke konjunkci lze vždy přidat **true** a k disjunkci **false**!)

Rezoluce: Rezolvujeme např. $\neg v \vee u$, $\neg u \vee \mathbf{false}$ na $\neg v \vee \mathbf{false}$:

$$T'' = \{\neg w \vee v, \; w \vee \mathbf{false}, \; \neg v \vee \mathbf{false}\}$$

(rezolventu umístíme na konec seznamu klauzulí)

a dále postupně:

$$T''' = \{w \vee \mathbf{false}, \neg w \vee \mathbf{false}\}$$

$$T'''' = \{\mathbf{false}\}$$

Odvodili jsme spor a tím dokázali, že $T \models u$.

Rezoluce a vyvracení (III)

Herbrandova věta: Množina formulí v klauzulárním tvaru je nesplnitelná právě tehdy, když existuje *konečná* množina jejich *základních* instancí, která je logicky sporná.
→ při rezoluci lze používat (unifikační) substituce.

SLD rezoluce (v Prologu): rezoluce na definitních klauzulích, s výběrovou (selection) funkcí, s lineární strategií.

S → pracuje se vždy s 1 rodičovským párem

L → jedním z členů páru je rezolventa z minulého kroku (→ prohledávání do hloubky); výběr první unifikovatelné klauzule z databáze

D → doplňkovým literálem druhého členu je jeho hlava

SLD rezoluce je *korektní*

$$\forall F, G \quad F \models G \quad if \quad F \vdash_{SLD} G$$

ale není úplná (problém nekonečného cyklu).

Rezoluce a vyvracení (IV)

Příklad: dotaz ?-syn(X,Y) a databáze
syn(X,Y) :- otec(Y,X),muz(X).
muz(karel˙IV).
otec(jan˙lucembursky,karel˙IV).

$$\{s(X_1, Y_1) \vee \overline{o(Y_1, X_1)} \vee \overline{m(X_1)},\ m(k),\ o(j, k)\} \cup \{\overline{s(k, X_0)}\}$$

$$\theta_0 = \{X_1/k,\ Y_1/X_0\}$$

$$\{s(k, X_0) \vee \overline{o(X_0, k)} \vee \overline{m(k)},\ m(k),\ o(j, k),\ \overline{s(k, X_0)}\}$$

$$\{\overline{o(X_0, k)} \vee \overline{m(k)},\ m(k),\ o(j, k)\}$$

$$\theta_1 = \{X_0/j\}$$

$$\{\overline{o(j, k)} \vee \overline{m(k)},\ m(k),\ o(j, k)\}$$

$$\{\overline{m(k)},\ m(k)\}$$

$$\square$$

Induktivní logické programování

Verze zima 1998

Induktivní logické programování

Upraveno podle *Lavrač, N. - Džeroski, S.: Inductive Logic Programming. Ellis Horwood 1994.*

ILP lze chápat v kontextu několika disciplín...

Logické programování : ILP je automatické *generování programů* (intenzionálních definic predikátů) z příkladů (extenzionálních definic).

Strojové učení (ML): ILP je strojové učení v *jazyce 1.řádu*.

Získávání znalostí z databází (KDD): ILP je získávání znalostí z relačních databází *s několika propojenými tabulkami*.

ILP – příklad

Hledáme intenzionální definici predikátu dcera/2 na základě **příkladů**

dcera(marketa,eliska_premyslovna).	$\oplus$
dcera(katerina,karel_IV).	$\oplus$
dcera(karel_IV,eliska_premyslovna).	$\ominus$
dcera(katerina,eliska_premyslovna).	$\ominus$

a **apriorních znalostí**

rodic(eliska_premyslovna,marketa).
rodic(eliska_premyslovna,karel_IV).
rodic(karel_IV,katerina).
rodic(karel_IV,vaclav_IV).
zena(eliska_premyslovna).
zena(marketa).
zena(katerina).

Správný tvar je:

dcera(X,Y) :- zena(X), rodic(Y,X).

ILP – obecný tvar úlohy

Dáno:

- množina *trénovacích příkladů* $\mathcal{E}$, složená z pozitivních příkladů $\mathcal{E}^+$ a z negativních příkladů $\mathcal{E}^-$; všechny příklady jsou základními fakty neznámého predikátu p;
- jazyk $\mathcal{L}$, určující *syntaktická omezení*, která musí definice predikátu p splňovat;
- množina *apriorních znalostí* ("background knowledge") $\mathcal{B}$, tj. množina definic predikátů q_i (různých od p); tyto predikáty mohou být využity v definici p.

Hledáme:

definici $\mathcal{H}$ pro predikát p, vyjádřenou pomocí $\mathcal{L}$; $\mathcal{H}$ musí být *úplná* a *konzistentní* vzhledem k příkladům $\mathcal{E}$ a apriorním znalostem $\mathcal{B}$.

ILP – pokrytí, úplnost, konzistence

Hypotéza $\mathcal{H}$ **pokrývá** příklad $e \in \mathcal{E}$ vzhledem k apriorním znalostem $\mathcal{B}$, jestliže $\mathcal{B} \cup \mathcal{H} \models e$ (tj. pravdivost příkladu logicky vyplývá ze současné pravdivosti hypotézy a apriorních znalostí).

Hypotéza $\mathcal{H}$ je **úplná** vzhledem k množině pozitivních a negativních příkladů $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ a apriorním znalostem $\mathcal{B}$, jestliže pokrývá všechny pozitivní příklady $e \in \mathcal{E}^+$ vzhledem k apriorním znalostem $\mathcal{B}$.

Hypotéza $\mathcal{H}$ je **konzistentní** vzhledem k množině pozitivních a negativních příkladů $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ a apriorním znalostem $\mathcal{B}$, jestliže nepokrývá žádný z negativních příkladů $e \in \mathcal{E}^+$ vzhledem k apriorním znalostem $\mathcal{B}$.

θ-subsumpce (Plotkin 1969)

Nechť c a c' jsou dvě programové klauzule. Klauzule c θ-**zahrnuje** klauzuli c' jestliže existuje substituce θ taková, že $c\theta \subseteq c'$.

θ-subsumpce jako relace **generalizace**: jestliže c θ-zahrnuje c', pak c je generalizací c' (tj. c je alespoň tak obecná jako c').

Vlastnosti θ-subsumpce:

- jestliže c θ-zahrnuje c', pak $c \models c'$;
- θ-subsumpce vytváří na množině *redukovaných* klauzulí (generalizační) *svaz*; tudíž každé dvě klauzule mají jednoznačně určenou *nejmenší horní mez* a *největší dolní mez*.

θ-subsumpce – příklad

Klauzule

$$\begin{aligned} c &= dcera(X,Y) \leftarrow rodic(Y,X) \\ &= \{dcera(X,Y), \overline{rodic(Y,X)}\} \end{aligned}$$

θ-zahrnuje klauzule:

$$\begin{aligned} c' &= dcera(X,Y) \leftarrow zena(X), rodic(Y,X) = \\ &= \{dcera(X,Y), \overline{zena(X)}, \overline{rodic(Y,X)}\} \end{aligned}$$

za identické substituce $\theta = \emptyset$;

$$c'' = dcera(X,X) \leftarrow zena(X), rodic(X,X)$$

za substituce $\theta = \{Y/X\}$;

$$\begin{aligned} c''' = \; & dcera(marketa, eliska) \leftarrow zena(marketa), \\ & rodic(eliska, marketa), rodic(eliska, karel) \end{aligned}$$

za substituce $\theta = \{X/marketa, Y/eliska\}$;

$$c'''' = dcera(X,Y) \leftarrow rodic(Y,X), rodic(W,V)$$

opět za identické substituce.

c a c'''' se θ-zahrnují *vzájemně*, protože naopak $c''''\theta$-zahrnuje c za substituce $\theta = \{W/Y,\ V/X\}$!

Nejméně obecná generalizace (I)

Nejmenší horní mez dvou redukovaných klauzulí c a c' ve svazu daném θ-subsumpcí se nazývá **nejméně obecná generalizace** ("least-general generalization"), $lgg(c, c')$.

Výpočet lgg:

lgg **termů:**

- pro dva identické termy: $lgg(t, t) = t$;
- lgg dvou různých konstant nebo struktur s různým funktorem nebo aritou je *proměnná*;
- lgg dvou struktur se stejným funktorem a aritou je struktura, jejíž argumenty jsou lgg dvojic argumentů, které mají v původních strukturách stejné pořadí;

lgg stejných podvýrazů (termů, atomů i literálů) se nahrazují stejnou proměnnou!

Příklady:

$$lgg([a, b, c], [a, c, d]) = [a, X, Y]$$

$$lgg(f(a, b), f(b, a)) = f(V, V)$$

Nejméně obecná generalizace (II)

lgg **atomů:**

- lgg dvou atomů se stejným predikátovým symbolem a aritou je atom, jehož argumenty jsou lgg dvojic argumentů, které mají v původních atomech stejné pořadí;
- lgg dvou atomů s různým predikátovým symbolem nebo aritou *není definována*;

lgg **literálů:**

- lgg dvojice pozitivních literálů - viz lgg atomů;
- lgg dvojice negativních literálů je negace lgg jejich atomů;
- lgg pozitivního a negativního literálu *není definována*;

Příklady:

$$lgg(rodic(eliska, marketa), rodic(eliska, karel)) = \\ = rodic(eliska, X)$$

$$lgg(rodic(eliska, marketa), \overline{rodic(eliska, karel)})$$
není definováno

Nejméně obecná generalizace (III)

lgg **klauzulí** je klauzule, jejímiž literály jsou lgg *všech dvojic* literálů z původních klauzulí, pro které je lgg definována.

Příklady:

Nechť

$$\begin{aligned} c_1 &= dcera(marketa, eliska) \leftarrow \\ &\quad zena(marketa), rodic(eliska, marketa) \\ c_2 &= dcera(katerina, karel) \leftarrow \\ &\quad zena(katerina), rodic(karel, katerina) \end{aligned}$$

Potom

$$lgg(c_1, c_2) = dcera(X, Y) \leftarrow zena(X), rodic(Y, X)$$

kde

$$X = lgg(marie, katerina) \quad Y = lgg(eliska, karel)$$

Nejméně obecná generalizace (IV)

Nechť

$$\begin{aligned} c_1 &= deda(jan, vaclav) \leftarrow \\ &\quad otec(jan, karel), otec(karel, vaclav) \\ c_2 &= deda(jan, jost) \leftarrow \\ &\quad otec(jan, jan\_jindrich), otec(jan\_jindrich, jost) \end{aligned}$$

Potom

$$\begin{aligned} lgg(c_1, c_2) &= deda(jan, X) \leftarrow otec(jan, Y), otec(Z, V), \\ &\quad otec(W, T), otec(Y, X) \end{aligned}$$

kde

$$\begin{array}{ll} X = lgg(vaclav, jost) & Y = lgg(karel, jan\_jindrich) \\ Z = lgg(jan, jan\_jindrich) & V = lgg(karel, jost) \\ W = lgg(karel, jan) & T = lgg(vaclav, jan\_jindrich) \end{array}$$

?Ale jak aplikovat lgg na původní úlohu:

- příklady ve formě základních atomů
- apriorní znalosti

Relativní lgg

Mějme dva příklady (základní atomy) e_1 a e_2, a množinu apriorních znalostí (základních faktů) $\mathcal{B}$. **Relativní nejméně obecná generalizace** e_1 a e_2 vzhledem k $\mathcal{B}$ je

$$\begin{aligned} rlgg(e_1, e_2, \mathcal{B}) &= lgg((e_1 \leftarrow c(\mathcal{B})), (e_2 \leftarrow c(\mathcal{B}))) = \\ &= lgg(e_1, e_2) \leftarrow lgg(c(\mathcal{B}), c(\mathcal{B})) \end{aligned}$$

kde $c(\mathcal{B})$ označuje konjunkci všech faktů z $\mathcal{B}$.

Problém: vznik nadbytečných literálů

V původním příkladu:

$$rlgg(e_1, e_2, \mathcal{B}) =$$

$$\mathbf{dcera}(\mathbf{X}, \mathbf{Y}) \leftarrow rodic(eliska, marketa), rodic(eliska, karel), rodic(karel, katerina), rodic(karel, vaclav), zena(eliska), zena(marketa), zena(katerina), rodic(eliska, Z), \mathbf{rodic}(\mathbf{Y}, \mathbf{X}), rodic(Y, Z), rodic(Y, V), rodic(Y, W), rodic(karel, T), zena(U), zena(S), \mathbf{zena}(\mathbf{X})$$

kde

$X = lgg(marketa, katerina)$ $\quad Y = lgg(eliska, karel)$
$Z = lgg(marketa, karel)$ $\quad V = lgg(marketa, vaclav)$
$W = lgg(karel, vaclav)$ $\quad T = lgg(katerina, vaclav)$
$U = lgg(eliska, marketa)$ $\quad S = lgg(eliska, katerina)$

Obrácení rezoluce (I)

Z klauzulí

$$\begin{aligned} b_1 &= zena(marketa) \\ b_2 &= rodic(eliska, marketa) \\ c &= dcera(X, Y) \leftarrow zena(X), rodic(Y, X) \end{aligned}$$

lze rezolučním principem s využitím substituce

$$\theta_1 = \{X/marketa\}$$

odvodit

$$\begin{aligned} res(b_1, c) &= dcera(marketa, Y) \\ &\leftarrow rodic(Y, marketa) \end{aligned}$$

a dále s využitím substituce

$$\theta_2 = \{Y/eliska\}$$

odvodit

$$res(b_2, res(b_1, c)) = dcera(marketa, eliska)$$

Obrácení rezoluce (II)

Tento postup lze obrátit: chceme nalézt hypotézu, která by společně s apriorními znalostmi

$$\begin{aligned} b_1 &= zena(marketa) \\ b_2 &= rodic(eliska, marketa) \end{aligned}$$

umožňovala odvodit příklad

$$e = dcera(marketa, eliska)$$

Pomocí **obrácené rezoluce** (Muggleton 1988) s využitím obrácené substituce

$$\theta_1^{-1} = \{eliska/Y\}$$

nalezneme hypotézu

$$c_1 = dcera(marketa, Y) \leftarrow rodic(Y, marketa)$$

a tu dále s využitím obrácené substituce

$$\theta_2^{-1} = \{marketa/X\}$$

zobecníme na

$$c = dcera(X, Y) \leftarrow zena(X), rodic(Y, X)$$

Obrácení rezoluce (III)

Problémy: obrácená rezoluce

- na rozdíl od rezoluce *není korektní*
  to platí obecně pro *induktivní* odvozovací pravidla
- je *nedeterministická*
  - výběr klauzulí
  - přiřazení proměnných při obrácené substituci

Příklad: chceme nalézt hypotézu, která by společně s apriorní znalostí $b = syn(vaclav, karel)$ umožňovala odvodit příklad $e = kral(vaclav)$. ?Jak zvolit mezi obrácenými substitucemi

$\theta_1^{-1} = \emptyset$
$res_1^{-1}(b, e) = kral(vaclav) \leftarrow syn(vaclav, karel)$

$\theta_2^{-1} = \{vaclav/X\}$
$res_2^{-1}(b, e) = kral(X) \leftarrow syn(X, karel)$

$\theta_3^{-1} = \{vaclav/X, karel/Y\}$
$res_3^{-1}(b, e) = kral(X) \leftarrow syn(X, Y)$

Specializační techniky (I)

Prohledávání stavového prostoru hypotéz (θ-subsumpčního svazu):

- **Generalizační** (bottom-up) techniky postupují zdola od příkladů;
  (nejméně obecná generalizace; obrácená rezoluce)

- **Specializační** (top-down) techniky postupují shora od nejobecnější definice cílového predikátu, a postupně ji zjemňují přidáváním literálů do těla (ev. substitucí za proměnné); typicky se přidávají literály

  – predikátů z apriorních znalostí;

  – elementárních predikátů (např. rovnost)

  s proměnnými z hlavy klauzule, nebo nejvýše 1 novou proměnnou.

a opět specializujeme. Máme k dispozici možnosti

$$c_{11} = dcera(X,Y) \leftarrow zena(X), rodic(Y,X)$$
$$c_{12} = dcera(X,Y) \leftarrow zena(X), rodic(Z,X)$$
$$c_{13} = dcera(X,Y) \leftarrow zena(X), rodic(Y,Z)$$

ovšem pouze c_{11} vyloučí zbylý $\ominus$. Výsledná úplná a konzistentní hypotéza tudíž je

$$H_2 = \{dcera(X,Y) \leftarrow zena(X), rodic(Y,X)\}$$

Specializační metody se používají zvláště při induktivním učení *z dat zatížených šumem* - pak se nepožaduje dokonalá úplnost a konzistence (namísto toho kritéria z teorie informace - entropie, informační zisk...).

Specializační techniky (II)

V příkladu začneme s nepodmíněnou klauzulí:

$$H_0 = \{c = dcera(X,Y) \leftarrow\}$$

Hypotéza pokrývá oba $\oplus$, ovšem i oba $\ominus$:

$$\begin{array}{ll} dcera(marketa, eliska) & \oplus \\ dcera(katerina, karel) & \oplus \\ dcera(karel, eliska) & \ominus \\ dcera(katerina, eliska) & \ominus \end{array}$$

Specializovat lze např. přidáním některého z literálů:

- $X = Y$, $zena(X)$, $zena(Y)$, $rodic(X,X)$, $rodic(X,Y)$, $rodic(Y,X)$, $rodic(Y,Y)$;
- $rodic(X,Z)$, $rodic(Z,X)$, $rodic(Y,Z)$, $rodic(Z,Y)$.

Z povolených minimálních specializací pouze

$$c_1 = dcera(X,Y) \leftarrow zena(X)$$
$$c_2 = dcera(X,Y) \leftarrow rodic(Y,X)$$
$$c_3 = dcera(X,Y) \leftarrow rodic(Z,X)$$
$$c_4 = dcera(X,Y) \leftarrow rodic(Y,Z)$$

stále pokrývají oba $\oplus$; c_3 a c_4 ovšem opět pokrývají oba $\ominus$, zatímco c_1 a c_2 pokrývají vždy jen jediný z $\ominus$. Zvolíme tedy např.

$$H_1 = \{c_1 = dcera(X,Y) \leftarrow zena(X)\}$$

ILP a relační databáze