Predikátová logika II. Úvod do logického programování. Intensionální logiky

Luboš Popelínský

E-mail: popel@fi.muni.cz
http://nlp.fi.muni.cz/uui/

Obsah:

- Normální formy v predikátové logice
- Skolemizace a unifikace
- Rezoluce v predikátové logice
- Logické programování
- Intensionální logiky

Obsah

Axiomatický systém predikátové logiky I

- používáme spojky $\{\Rightarrow, \neg\}$ a kvantifikátor $\forall$. Ostatní spojky jsou chápány jako zkratky uvedené ve výr. logice, resp. $\exists xA =_{df} \neg\forall x\neg A$
- axiomy pro spojky (A, B, C jsou formule) – shodné s výr. logikou:
  A$_1$ $A \Rightarrow (B \Rightarrow A)$
  A$_2$ $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
  A$_3$ $(\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$
- axiomy pro $\forall$:
  A$_4$ $\forall xA \Rightarrow A(x/t)$
  A$_5$ $\forall x(A \Rightarrow B) \Rightarrow (A \Rightarrow \forall xB), A$ neobsahuje volně x

Axiomatický systém predikátové logiky II

- axiomy pro rovnost:

  A$_6$ $x = x$

  A$_7$ $(x = y) \Rightarrow (f(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n))$

  A$_8$ $(x = y) \Rightarrow (P(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) = P(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n))$

- odvozovací pravidla:
  - modus ponens (MP, stejné s výr. logikou): z A a $A \Rightarrow B$ odvodíme B pro libovolné formule A, B
  - generalizace (PG): z A odvodíme $\forall x A$

Axiomatický systém: příklad

Příklad: důkaz z předpokladu. Ukažte, že pokud je dokazatelná formule $A \Rightarrow B$ a proměnná x není volná v A, pak je dokazatelná i formule $A \Rightarrow \forall x B$.

1.	$\vdash A \Rightarrow B$	předpoklad
2.	$\vdash \forall x(A \Rightarrow B)$	PG(1)
3.	$\vdash \forall x(A \Rightarrow B) \Rightarrow (A \Rightarrow \forall x B)$	$\mathbf{A_5}$
4.	$\vdash A \Rightarrow \forall x B$	MP(2,3)

Využití axiomatického systému: teorie

- teorie: množina uzavřených formulí **T**, model teorie **T**: interpretace, v níž je každá formule z **T** pravdivá

Příklad: teorie elementární aritmetiky (0, unární funkce následník s, binární $+, *$)

- **Ax₁** $\forall x \neg(s(x) = 0)$
  Ax₂ $\forall x (x + 0 = x)$
  Ax₃ $\forall x \forall y (x + s(y) = s(x + y))$
  Ax₄ $\forall x (x * 0 = 0)$
  Ax₅ $\forall x \forall y (x * s(y) = (x * y) + x)$
- pomocná odvozovací pravidla:
  (PS) je-li $\vdash x = y$, pak $\vdash y = x$ (symetrie $=$)
  (PT) je-li $\vdash x = y$ a $\vdash y = z$, pak $\vdash x = z$ (tranzitivita $=$)
  (PPS) je-li $\vdash x = y$, pak $\vdash s(x) = s(y)$ (přidání s)
  (PVS) je-li $\vdash s(x) = s(y)$, pak $\vdash x = y$ (vypuštění s)

Teorie: příklad důkazu

Příklad: dokažte v teorii elementární aritmetiky $s(0) + s(0) = s(s(0))$.

1.	$\vdash \forall x \forall y (x + s(y) = s(x + y))$	$\mathbf{Ax_3}$
2.	$\vdash (\forall x \forall y (x + s(y) = s(x + y))) \Rightarrow$	
	$(\forall y (s(0) + s(y) = s(s(0) + y)))$	$\mathbf{A_4}$
3.	$\vdash \forall y (s(0) + s(y) = s(s(0) + y))$	MP(1,2)
4.	$\vdash (\forall y (s(0) + s(y) = s(s(0) + y))) \Rightarrow$	
	$(s(0) + s(0) = s(s(0) + 0))$	$\mathbf{A_4}$
5.	$\vdash s(0) + s(0) = s(s(0) + 0)$	MP(3,4)
6.	$\vdash \forall x (x + 0 = x)$	$\mathbf{Ax_2}$
7.	$\vdash (\forall x (x + 0 = x)) \Rightarrow (s(0) + 0 = s(0))$	$\mathbf{A_4}$
8.	$\vdash s(0) + 0 = s(0)$	MP(6,7)
9.	$\vdash s(s(0) + 0) = s(s(0))$	PPS(8)
10.	$\vdash s(0) + s(0) = s(s(0))$	PT(5,9)

Normální formy v predikátové logice

Prenexová normální forma (pnf)

- cíl: převést libovolnou (uzavřenou) formuli do tvaru, v němž jsou všechny kvantifikátory na začátku a následuje otevřené (= bez kvantifikátorů) jádro v nkf (ndf)

  $$Qx_1 \dots Qx_n((A_{1_1} \vee \dots \vee A_{1_{l_1}}) \wedge (A_{2_1} \vee \dots \vee A_{2_{l_2}}) \wedge \dots \wedge (A_{m_1} \vee \dots \vee A_{m_{l_m}}))$$
- příklad: $\forall x \forall y \exists z \forall w((P(x,y) \vee \neg Q(z)) \wedge (R(x,w) \vee R(y,w)))$
- **Věta:** pro každou formuli existuje ekvivalentní formule v konjunktivní (disjunktivní) prenexové normální formě.

Prenexová normální forma: algoritmus převodu

1. eliminovat zbytečné kvantifikátory
2. přejmenovat korektně proměnné tak, aby u každého kvantifikátoru byla jiná proměnná
3. eliminovat všechny spojky různé od $\neg, \wedge$ a $\vee$
4. přesunout negaci dovnitř, je-li potřeba:
   $\neg\forall x A$ nahradit $\exists x \neg A$
   $\neg(A \wedge B)$ nahradit $\neg A \vee \neg B$ apod.
5. přesunout kvantifikátory doleva ($o \in \{\wedge, \vee\}$, $Q \in \{\forall, \exists\}$):
   $A \ o \ QxB$ nahradit $Qx(A \ o \ B)$
   $QxA \ o \ B$ nahradit $Qx(A \ o \ B)$
6. použít distributivní zákony k převodu jádra do nkf (ndf):
   $A \vee (B \wedge C)$ nahradit $(A \vee B) \wedge (A \vee C)$
   $(A \wedge B) \vee C$ nahradit $(A \vee C) \wedge (B \vee C)$

Převod do pnf – příklad

Příklad: převeďte do konjunktivní pnf formuli
$\forall x \exists y \neg(P(x,y) \Rightarrow \forall z R(y)) \vee \neg\exists x Q(x)$.

1. $\forall x \exists y \neg(P(x,y) \Rightarrow R(y)) \vee \neg\exists x Q(x)$ (zbytečné $\forall z$)
2. $\forall x_1 \exists y \neg(P(x_1,y) \Rightarrow R(y)) \vee \neg\exists x_2 Q(x_2)$ (přejmenování x)
3. $\forall x_1 \exists y \neg(\neg P(x_1,y) \vee R(y)) \vee \neg\exists x_2 Q(x_2)$ (eliminace $\Rightarrow$)
4. $\forall x_1 \exists y (P(x_1,y) \wedge \neg R(y)) \vee \forall x_2 \neg Q(x_2)$ (přesun negace 2x)
5. $\forall x_1 (\exists y (P(x_1,y) \wedge \neg R(y)) \vee \forall x_2 \neg Q(x_2))$ (posun $\forall x_1$ doleva)
   $\forall x_1 \exists y ((P(x_1,y) \wedge \neg R(y)) \vee \forall x_2 \neg Q(x_2))$ (posun $\exists y$ doleva)
   $\forall x_1 \exists y \forall x_2 ((P(x_1,y) \wedge \neg R(y)) \vee \neg Q(x_2))$ (posun $\forall x_2$ doleva)
6. $\forall x_1 \exists y \forall x_2 ((P(x_1,y) \vee \neg Q(x_2)) \wedge (\neg R(y) \vee \neg Q(x_2)))$ (jádro do nkf)

Skolemizace a unifikace

potřebujeme mj. pro rezoluci

Skolemizace

- převod formulí na formule bez existenčních kvantifikátorů v jazyce, který je rozšířen o tzv. Skolemovy funkce; zachovává splnitelnost
- idea převodu: formuli $\forall x_1 \ldots \forall x_n \exists y P(x_1, \ldots, x_n, y)$ transformujeme na $\forall x_1 \ldots \forall x_n P(x_1, \ldots, x_n, f(x_1, \ldots, x_n))$
- příklad: mějme celá čísla s $+$. Formuli $\forall x \exists y (x + y = 0)$ převedeme na $\forall x (x + f(x) = 0)$. Interpretace f – unární funkce, která pro daný argument vrátí opačné číslo.
- Skolemova normální forma je prenexová normální forma pouze s univerzálními kvantifikátory.
- **Věta:** každou formuli A lze převést na takovou formuli A' ve Skolemově normální formě, že A je splnitelná právě když A' je splnitelná.

Algoritmus převodu do Skolemovy nf

1. převést formuli do prenexové konjunktivní nf
2. provést Skolemizaci: odstranit všechny existenční kvantifikátory a nahradit jimi vázané proměnné pomocnými Skolemovými funkcemi

- příklad 1: převeďte do Skolemovy nf formuli
  $\forall x \exists y \neg(P(x,y) \Rightarrow \forall z R(y)) \vee \neg\exists x Q(x)$
  1. $\forall x_1 \exists y \forall x_2((P(x_1,y) \vee \neg Q(x_2)) \wedge (\neg R(y) \vee \neg Q(x_2)))$
  2. $\forall x_1 \forall x_2((P(x_1,f(x_1)) \vee \neg Q(x_2)) \wedge (\neg R(f(x_1)) \vee \neg Q(x_2)))$
- příklad 2: převeďte do Skolemovy nf následující formuli v pnf
  $\forall x \exists y \forall z \exists w (P(x,y) \vee \neg Q(z,w))$
  2. $\forall x \forall z (P(x, f_1(x)) \vee \neg Q(z, f_2(x,z)))$

Herbrandova věta I

- motivace: hledáme snazší prostředky k určení, zda daná množina formulí je splnitelná
- pracujeme s množinou S formulí ve Skolemově nf (univerzální kvantifikátory se často při zápisu vynechávají), jejími konstantami (alespoň jedna, příp. přidaná mimo S), funkčními a predikátovými symboly
- Herbrandovo univerzum $U(S)$ je množina všech uzavřených termů, které lze utvořit z konstant a funkčních symbolů z S (tzv. základní termy)
  př.: pro $S = \{P(f(0))\}$ je $U(S) = \{0, f(0), f(f(0)), f(f(f(0))), \dots\}$
- Herbrandova báze $B(S)$ je množina všech atomických formulí, které lze vytvořit nad prvky $U(S)$;
  $B(S) = \{P(t_1, \dots, t_n) | t_i \in U(S), P \text{ je predik. symbol figurující v } S\}$
  př.: pro $S = \{P(f(0))\}$ je $B(S) = \{P(0), P(f(0)), P(f(f(0))), \dots\}$

Herbrandova věta II

- Herbrandova interpretace je libovolná podmnožina báze $B(S)$ zahrnující ty aplikace predikátů na prvky univerza, které jsou pravdivé
  Poznámka: s funkcemi a konstantami lze pracovat i nadále pouze na symbolické úrovni
- Herbrandův model $M(S)$ množiny S je taková Herbrandova interpretace, ve které jsou všechny formule z S pravdivé
- **Herbrandova věta:** buď existuje Herbrandův model S nebo existuje konečně mnoho uzavřených instancí prvků S, jejichž konjunkce neplatí
- slabší tvrzení: S je splnitelná právě tehdy, když existuje její Herbrandův model
- závěr: k rozhodnutí o splnitelnosti množiny již nepotřebujeme brát v úvahu všechny možné interpretace, stačí pracovat pouze se ‚symbolickými' Herbrandovými interpretacemi

Herbrandovy modely – příklady

Příklad 1: $S = \{P(0), P(s(x)) \vee \neg P(x)\}$ (předp. $\forall$ kvantifikovány)

- $U(S) = \{0, s(0), s(s(0)), s(s(s(0))), \ldots\}$
  $B(S) = \{P(0), P(s(0)), P(s(s(0))), P(s(s(s(0)))), \ldots\}$
  $M(S) = B(S)$ (minimální Herbrandův model je celá báze)
  poznámka: P vyjadřuje vlastnost ‚být korektní přirozené číslo' (pomocí následníků nuly)

Příklad 2: $S' = \{P(0), P(s(x)) \vee \neg P(x), R(x, s(x)) \vee \neg P(x)\}$

- $U(S') = \{0, s(0), s(s(0)), s(s(s(0))), \ldots\}$ (stejné jako pro S)
  $B(S') = \{P(0), P(s(0)), P(s(s(0))), P(s(s(s(0)))), \ldots,$
  $R(0, 0), R(0, s(0)), R(s(0), 0), R(s(0), s(0)), \ldots\}$
  $M(S') = \{P(0), P(s(0)), P(s(s(0))), P(s(s(s(0)))), \ldots,$
  $R(0, s(0)), R(s(0), s(s(0))), R(s(s(0)), s(s(s(0)))) \ldots\}$
  poznámka: P je stejné jako pro S, $R(x, y)$ reprezentuje binární vlastnost ‚y je následníkem x' (resp. ‚x je předchůdcem y')

Unifikace – motivace

- směřujeme k rezoluci v predikátové logici:
  - formule umíme reprezentovat v konjunktivní pnf odpovídající klauzulární formě ($\forall$ nepíšeme, ale předpokládáme univerzální kvantifikaci všech proměnných)
  - literály nyní představují atomické formule a jejich negace
  - zůstává jediný problém: jak instanciovat proměnné, aby bylo možné použít rezoluční pravidlo
- příklad: mějme klauzule $C_1 = \{P(f(x)), \neg Q(a,x)\}$ a $C_2 = \{\neg P(f(g(a)))\}$; nahradíme-li x termem $g(a)$, získáme rezolventu $\{\neg Q(a, g(a))\}$
  poznámka: C_1 odpovídá formuli $\forall x(P(f(x)) \vee \neg Q(a,x))$, takže můžeme použít libovolnou instanci
- obecně řeší uvedený problém se substitucemi proměnných unifikace

Substituce

- konečná substituce ϕ je konečná množina $\{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$, kde všechna x_i jsou vzájemně různé proměnné a každé t_i je term různý od x_i. Jsou-li všechna t_i uzavřené termy, jedná se o uzavřenou substituci. Pokud jsou t_i proměnné, označujeme ϕ jako přejmenování proměnných.
- označme libovolný term nebo literál jako výraz E; pak $E\phi$ je výsledek nahrazení všech výskytů všech x_i odpovídajícími termy t_i (obdobně pro množiny výrazů)
- poznámka: substituce proměnných probíhají paralelně, ne postupně
- příklad:
  $S = \{f(x, g(y)), \neg P(y, x), Q(y, z, a)\}$
  $\phi = \{x/h(y), y/g(z), z/c\}$
  $S\phi = \{f(h(y), g(g(z))), \neg P(g(z), h(y)), Q(g(z), c, a)\}$

Kompozice substitucí

- kompozice substitucí
  $\phi = \{x_1/t_1, \ldots, x_n/t_n\}$ a $\psi = \{y_1/s_1, \ldots, y_m/s_m\}$ je množina
  $\phi\psi = \{x_1/t_1\psi, \ldots, x_n/t_n\psi, y_1/s_1, \ldots, y_m/s_m\}$ beze všech $x_i/t_i\psi$, pro která $x_i = t_i\psi$, a všech $y_j/s_j, y_j \in \{x_1, \ldots, x_n\}$
- pro prázdnou substituci ϵ a libovolnou substituci ϕ platí $\phi\epsilon = \epsilon\phi = \phi$
- pro libovolný výraz E a substituce ϕ, ψ, σ platí $(E\phi)\psi = E(\phi\psi)$ a $(\phi\psi)\sigma = \phi(\psi\sigma)$
- příklad:
  $S = \{f(x, g(y)), \neg P(y, x), Q(y, z, a)\}$
  $\phi = \{x/h(y), y/w, z/g(w, y)\}$, $\psi = \{x/a, y/f(b), w/y\}$
  $\phi\psi = \{x/h(f(b)), z/g(y, f(b)), w/y\}$
  $S\phi = \{f(h(y), g(w)), \neg P(w, h(y)), Q(w, g(w, y), a)\}$
  $S(\phi\psi) = (S\phi)\psi =$
  $\{f(h(f(b)), g(y)), \neg P(y, h(f(b))), Q(y, g(y, f(b)), a)\}$

Unifikace

- substituci ϕ nazveme unifikátorem množiny $S = \{E_1, \ldots, E_2\}$, pokud $E_1\phi = E_2\phi = \cdots = E_n\phi$, tedy v případě, že $S\phi$ má jediný prvek. Existuje-li unifikátor množiny, označíme ji jako unifikovatelnou.
- příklady:
  1. $\{P(x, a), P(b, c)\}, \{P(f(x), z), P(a, w)\}, \{P(x, w), \neg P(a, w)\}$, $\{P(x, y, z), P(a, b)\}, \{R(x), P(x)\}$ nejsou unifikovatelné
  2. unifikátorem $\{P(x, c), P(b, c)\}$ je $\phi = \{x/b\}$; žádný jiný neexistuje
  3. unifikátorem $\{P(f(x), y), P(f(a), w)\}$ může být $\phi = \{x/a, y/w\}$, ale také $\psi = \{x/a, y/a, w/a\}$, $\sigma = \{x/a, y/b, w/b\}$ atd.
- unifikátor ϕ množiny S je nejobecnějším unifikátorem (mgu) S, pokud pro libovolný unifikátor ψ množiny S existuje substituce λ taková, že $\phi\lambda = \psi$
- příklad: v bodu 3. předchozího příkladu je mgu ϕ, odpovídající substituce λ pro unifikátory ψ resp. σ je $\{w/a\}$ resp. $\{w/b\}$

Rozdíl mezi výrazy

- poznámka: pro unifikovatelné množiny existuje jediný mgu (až na přejmenování proměnných)
- mějme množinu S výrazů. Najdeme první (nejlevější) pozici, na které nemají všechny prvky S stejný symbol. Rozdíl $D(S)$ mezi výrazy pak definujeme jako množinu podvýrazů všech $E \in S$ začínajících na této pozici.
  Poznámka: každý unifikátor S nutně unifikuje i $D(S)$.
- příklady:
  - $S_1 = \{f(\mathbf{x}, g(x)), f(\mathbf{h}(y), g(h(z)))\}$
    $D(S_1) = \{x, h(y)\}$
    $T_1 = S_1\{x/h(y)\} = \{f(h(y), g(h(\mathbf{y}))), f(h(y), g(h(\mathbf{z})))\}$
    $D(T_1) = \{y, z\}$
  - $S_2 = \{f(\mathbf{h}(x), g(x)), f(\mathbf{g}(x), f(x))\}$
    $D(S_2) = \{h(x), g(x)\}$

Unifikační algoritmus

unifikační algoritmus pro množinu výrazů S

- krok 0: $S_0 := S, \phi_0 := \epsilon$
- krok $k+1$:
  - je-li S_k jednoprvková, ukonči algoritmus s výsledkem $mgu(S) = \phi_0\phi_1\phi_2 \ldots \phi_k$
  - jinak, pokud v $D(S_k)$ není proměnná v a term t, který ji neobsahuje, ukonči algoritmus s výsledkem neexistuje $mgu(S)$
  - jinak vyber takovou proměnnou v a term t, který neobsahuje v, $\phi_{k+1} := \{v/t\}$, $S_{k+1} := S_k\phi_{k+1}$ a pokračuj krokem $k+2$

Algoritmus skončí korektně pro libovolnou množinu výrazů S.

Unifikace – příklad

Najděte nejobecnější unifikátor množiny
$S = \{P(f(y, g(z)), h(b)), P(f(h(w), g(a)), x), P(f(h(b), g(z)), y)\}$

1. $S_0 = S$ není jednoprvková; $D(S_0) = \{y, h(w), h(b)\}$, vyberme ze dvou možností $\phi_1 = \{y/h(w)\}$, $S_1 = S_0\phi_1 =$
$\{P(f(h(w), g(z)), h(b)), P(f(h(w), g(a)), x), P(f(h(b), g(z)), h(w))\}$
2. $D(S_1) = \{w, b\}, \phi_2 = \{w/b\}, S_2 = S_1\phi_2 =$
$\{P(f(h(b), g(z)), h(b)), P(f(h(b), g(a)), x), P(f(h(b), g(z)), h(b))\}$
3. $D(S_2) = \{z, a\}, \phi_3 = \{z/a\}, S_3 = S_2\phi_3 =$
$\{P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), x), P(f(h(b), g(a)), h(b))\}$
4. $D(S_3) = \{h(b), x\}, \phi_4 = \{x/h(b)\}, S_4 = S_3\phi_4 =$
$\{P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), h(b))\}$
5. $mgu(S) = \{y/h(w)\}\{w/b\}\{z/a\}\{x/h(b)\} =$
$\{y/h(b), w/b, z/a, x/h(b)\}$

Rezoluce. Úvod

- metoda založená na vyvracení, pracujeme s formulemi ve Skolemově normální formě, kvantifikátory nepíšeme
- jako ve výrokové logice: formuli $\forall x \forall y ((P(x, f(x)) \vee \neg Q(y)) \wedge (\neg R(f(x)) \vee \neg Q(y)))$ reprezentujeme jako $\{\{P(x, f(x)), \neg Q(y)\}, \{\neg R(f(x)), \neg Q(y)\}\}$

Standardizace proměnných

- proměnné chápeme jako lokální v dané klauzuli (pozn.: $\forall x (A(x) \wedge B(x)) \Leftrightarrow (\forall x A(x) \wedge \forall x B(x)) \Leftrightarrow (\forall x A(x) \wedge \forall y B(y)))$
- mezi stejnými proměnnými v různých klauzulích není žádná vazba
- standardizace proměnných: přejmenujeme proměnné v různých klauzulích tak, aby v nich žádné stejně pojmenované proměnné nevystupovaly
- standardizace proměnných je nezbytná, příklad: $\{\{P(x)\}, \{\neg P(f(x))\}\}$ je nesplnitelná $\Leftrightarrow$ bez přejmenování $P(x)$ a $P(f(x))$ nezunifikujeme $\Leftrightarrow$ a bez unifikace nemůžeme rezolvovat

Rezoluční pravidlo pro predikátovou logiku

- rezoluční pravidlo pro predikátovou logiku: mějme klauzule C_1 a C_2 bez společných proměnných (po případném přejmenování) ve tvaru $C_1 = C_1' \sqcup \{P(\vec{x}_1), \ldots, P(\vec{x}_n)\}$, $C_2 = C_2' \sqcup \{\neg P(\vec{y}_1), \ldots, \neg P(\vec{y}_m)\}$. Je-li ϕ mgu množiny $\{P(\vec{x}_1), \ldots, P(\vec{x}_n), P(\vec{y}_1), \ldots, P(\vec{y}_m)\}$, pak rezolventou C_1 a C_2 je $C_1'\phi \cup C_2'\phi$.
- rezoluční důkazy a rezoluční vyvrácení definujeme stejně jako ve výrokové logice, pouze používáme rezoluční pravidlo pro predikátovou logiku:
  - rezoluční důkaz C z S je binární strom s listy z S a kořenem C, jehož každý vnitřní uzel je rezolventou svých bezprostředních následníků,
  - rezoluční vyvrácení S je rezoluční důkaz $\square$ z S

Rezolventa – příklady I

Př. 1: rezolvujeme klauzule $\{P(x, a)\}$ a $\{\neg P(x, x)\}$

- přejmenujeme proměnné např. v první klauzuli: $\{P(x_1, a)\}$
- $mgu(\{P(x_1, a), P(x, x)\}) = \{x_1/a, x/a\}$
- rezolventa $\square$

Př. 2: rezolvujeme klauzule $\{P(x, y), \neg R(x)\}$ a $\{\neg P(a, b)\}$

- $mgu(\{P(x, y), P(a, b)\}) = \{x/a, y/b\}$
- aplikujeme mgu na $\{\neg R(x)\}$
- rezolventa $\{\neg R(a)\}$

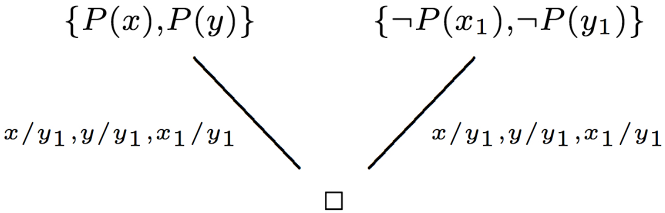
Rezolventa – příklady II

Př. 3: rezolvujeme klauzule $C_1 = \{Q(x), \neg R(y), P(x,y), P(f(z), f(z))\}$ a $C_2 = \{\neg N(u), \neg R(w), \neg P(f(a), f(a)), \neg P(f(w), f(w))\}$

- vybereme množinu literálů, na kterých budeme rezolvovat $\{P(x,y), P(f(z), f(z)), P(f(a), f(a)), P(f(w), f(w))\}$
- její mgu $\phi = \{x/f(a), y/f(a), z/a, w/a\}$
- $C_1' = \{Q(x), \neg R(y)\}$, $C_1'\phi = \{Q(f(a)), \neg R(f(a))\}$
- $C_2' = \{\neg N(u), \neg R(w)\}$, $C_2'\phi = \{\neg N(u), \neg R(a)\}$
- výsledná rezolventa $C_1'\phi \cup C_2'\phi = \{Q(f(a)), \neg R(f(a)), \neg N(u), \neg R(a)\}$

Rezoluční důkazy

- obecné schméma důkazu ‚A je důsledkem množiny formulí $\mathbf{T}$': všechny prvky $\mathbf{T}$ a $\neg A$ převedeme do klauzulární formy, dokazujeme nesplnitelnost jejich sjednocení
- poznámka: při jednom rezolučním kroku musíme být schopni odstranit několik literálů zároveň (pokud bychom v následujícím příkladu rezolvovali vždy pouze na jediném literálu, množinu rezolučně nevyvrátíme)
- příklad: ukažte rezoluční vyvrácení $\{\{P(x), P(y)\}, \{\neg P(x_1), \neg P(y_1)\}\}$



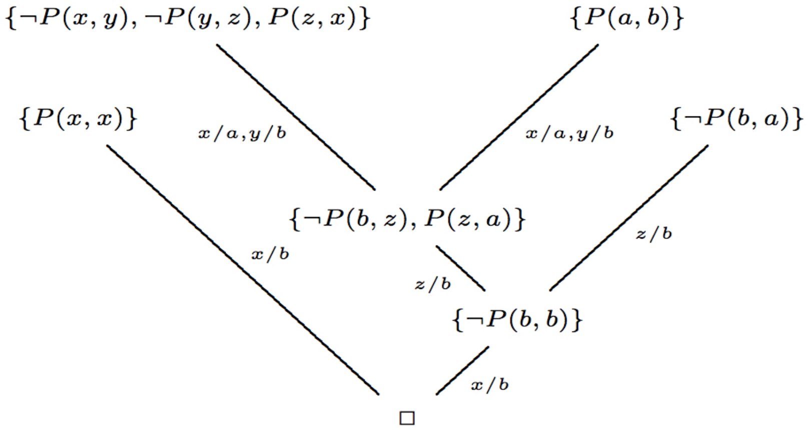
Rezoluční důkaz – příklad I

Z předpokladu reflexivity $\forall x P(x,x)$ a vlastnosti $\forall x \forall y \forall z((P(x,y) \wedge P(y,z)) \Rightarrow P(z,x))$ dokažte symetrii $\forall x \forall y(P(x,y) \Rightarrow P(y,x))$.

- převedeme předpoklady do klauzulárního tvaru:
  $S_1 = \{\{P(x,x)\}\}$
  $S_2 = \{\{\neg P(x,y), \neg P(y,z), P(z,x)\}\}$
- dokazovanou formuli negujeme:
  $\exists x \exists y(P(x,y) \wedge \neg P(y,x))$,
  převedeme do klauzulárního tvaru (přes Skolemovu nf):
  $P(a,b) \wedge \neg P(b,a)$
  $S = \{\{P(a,b)\}, \{\neg P(b,a)\}\}$
- dokazujeme nesplnitelnost $S_1 \cup S_2 \cup S$

Rezoluční důkaz – příklad II

- dokazujeme nesplnitelnost množiny klauzulí $\{\{P(x,x)\},\{\neg P(x,y),\neg P(y,z),P(z,x)\},\{P(a,b)\},\{\neg P(b,a)\}\}$
- strom rezolučního vyvrácení (jeden z možných):

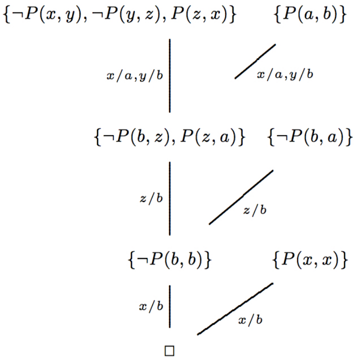


Rezoluce – vlastnosti

- stejně jako ve výrokové logice je rezoluce v predikátové logice korektní a úplná
- stejný problém jako ve výrokové logice: strategie generování rezolvent (příliš velký prohledávaný prostor)
- lze použít všechny metody zjemnění uvedené pro výrokovou logiku (T-rezoluce, sémantická rezoluce atd.)
- uvedeme pouze příklady variant rezoluce postupně směřujících k SLD-rezoluci (používané též v Prologu)

Lineární rezoluce

- lineární struktura důkazu, rezolvujeme vždy předchozí rezolventu s klauzulí z vyvracené množiny nebo dříve odvozenou rezolventou; korektní a úplná
- příklad: lineární rezoluční vyvrácení množiny $\{\{P(x,x)\},\{\neg P(x,y),\neg P(y,z),P(z,x)\},\{P(a,b)\},\{\neg P(b,a)\}\}$



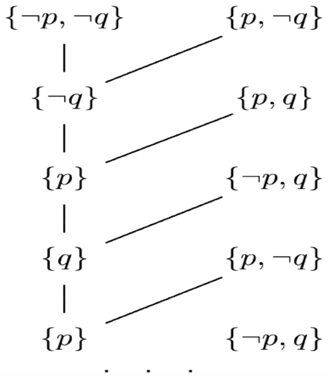
Hornovy klauzule

- omezení na určitý typ klauzulí používaných v programovacím jazyce Prolog
- Hornova klauzule je klauzule s nejvýše jedním pozitivním literálem.
  Příklad: $C = \{p, \neg q, \neg r\}$
  Běžný zápis ve výrokové logice $p \vee \neg q \vee \neg r$ lze ekvivalentními úpravami převést na $p \vee \neg(q \wedge r)$ a dále na $p \Leftarrow q \wedge r$, což v Prologu zapisujeme `p :- q, r.`
- typy Hornových klauzulí:
  - programové s právě jedním pozitivním literálem dále dělíme na
    * fakta bez negativních lit. (př.: $\{p\}$, v Prologu `p.`)
    * pravidla s alespoň jedním negativním lit. (př.: $\{p, \neg q\}$, `p :- q.`)
  - cíle bez pozitivního literálu (př.: $\{\neg p\}$, `:- p.` resp. `?- p.`)
- Každá nesplnitelná množina neprázdných Hornových klauzulí musí obsahovat alespoň jeden fakt a alespoň jeden cíl.

LI-rezoluce a LD-rezoluce

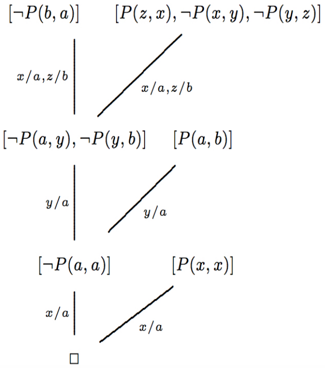
- LI-rezoluce - lineární rezoluce začínající cílovou klauzulí (žádný pozitivní literál), rezolvujeme vždy předchozí rezolventu (výhradně) s klauzulí z vyvracené množiny. Korektní, obecně není úplná; úplná pro Hornovy klauzule.
- LD-rezoluce vychází z LI-rezoluce, směřujeme k implementaci
- pracujeme výhradně s Hornovými klauzulemi (nejvýše jeden pozitivní literál)
- klauzule nahradíme uspořádanými klauzulemi; změna notace z $\{P(x), \neg R(x, f(y)), \neg Q(a)\}$ na $[P(x), \neg R(x, f(y)), \neg Q(a)]$
- rezoluce pro uspořádané klauzule v predikátové logice: mějme uspořádané klauzule bez společných proměnných (po případném přejmenování)
  $G = [\neg A_1, \neg A_2, \ldots, \neg A_n]$ a
  $H = [B_0, \neg B_1, \neg B_2, \ldots, \neg B_m]$,
  rezolventou G a H pro $\phi = mgu(B_0, A_i)$ bude uspořádaná klauzule
  $[\neg A_1\phi, \neg A_2\phi, \ldots, \neg A_{i-1}\phi, \neg B_1\phi, \neg B_2\phi, \ldots, \neg B_m\phi, \neg A_{i+1}\phi, \ldots, \neg A_n\phi]$

Neúplnost LI-rezoluce



LD-rezoluce: příklad

- LD-rezoluční vyvrácení množiny uspořádaných klauzulí $\{[P(x,x)], [P(z,x), \neg P(x,y), \neg P(y,z)], [P(a,b)], [\neg P(b,a)]\}$

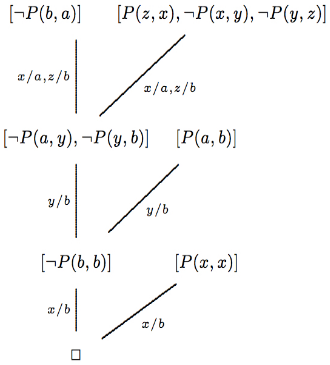


SLD-rezoluce

- pomocí selekčního pravidla algoritmizuje výběr literálu z cílové klauzule, na kterém se bude rezolvovat
- SLD-rezoluce (s libovolným selekčním pravidlem) je korektní a úplná pro Hornovy klauzule
- budeme používat selekční pravidlo, které vybírá nejlevější literál
- generování rezolvent pro uvedené pravidlo:
  $G = [\neg A_1, \neg A_2, \ldots, \neg A_n]$,
  $H = [B_0, \neg B_1, \neg B_2, \ldots, \neg B_m]$,
  rezolventou G a H pro $\phi = mgu(B_0, A_1)$ je
  $[\neg B_1\phi, \neg B_2\phi, \ldots, \neg B_m\phi, \neg A_2\phi, \ldots, \neg A_n\phi]$

SLD-rezoluce: příklad

- příklad: SLD-rezoluční vyvrácení se zvoleným selekčním pravidlem (vybírajícím vždy nejlevější literál)



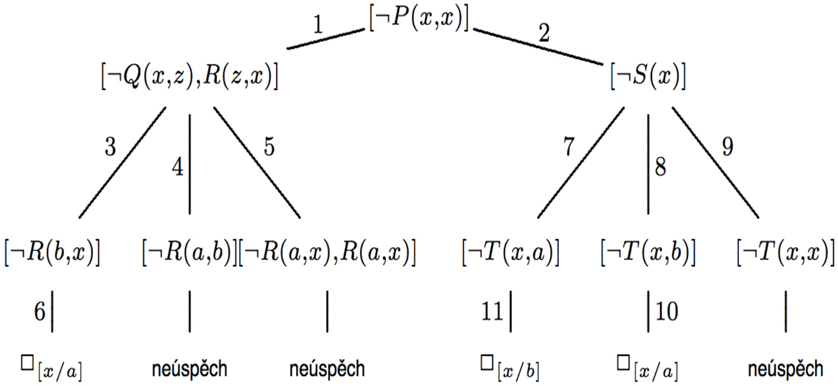
- srovnání s LD-rezolucí: ve druhém kroku musíme rezolvovat na $\neg P(a,y)$ (v LD-rezoluci bylo možné vybrat libovolný z literálů $\neg P(a,y), \neg P(y,b)$)

SLD-rezoluce: význam

- máme množinu programových klauzulí P a cílovou klauzuli $G = [\neg A_1(\vec{x}), \ldots, \neg A_n(\vec{x})]$
- dokazujeme nesplnitelnost $P \cup \{G\}$, tj. $P \wedge \forall \vec{x}(\neg A_1(\vec{x}) \vee \ldots \vee \neg A_n(\vec{x}))$
- uvedená nesplnitelnost je ekvivalentní $P \vdash \neg G$, tj. $P \vdash \exists \vec{x}(A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}))$
- zadáním cíle G tedy chceme zjistit, zda existují nějaké objekty (případně jaké), které na základě P splňují formuli $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x})$
- aplikujeme-li kompozici všech mgu postupně použitých při SLD-odvození na jednotlivé proměnné vektoru $\vec{x}$, získáme konkrétní příklady zmíněných objektů (termů) splňujících danou formuli

SLD-stromy

Prostor všech možných SLD-derivací při vyhodnocování daného cíle G pro program P dokážeme zachytit SLD-stromem. Příklad:



Logické programování

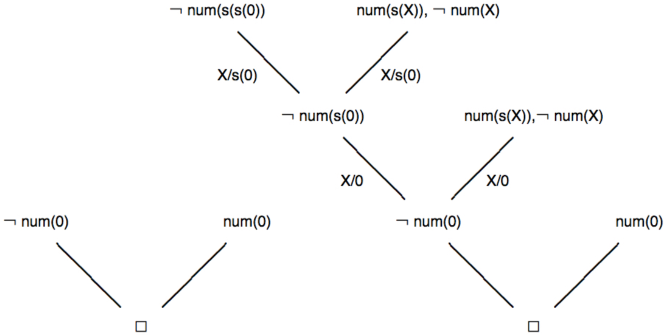
- strategie prohledávání stavového prostoru (SLD-stromu) do hloubky
- výběr programových klauzulí shora dolů
- výběr podcílů zleva doprava (viz selekční pravidlo v SLD rezoluci)
- silně využívá rekurzi
- historie: 70. l. 20. stol. – Colmerauer, Kowalski; D.H.D. Warren (WAM)
- jazyk Prolog

SLD-odvození pro logický program

Přirozená čísla s nulou $0 \in N \wedge \forall X : ((X+1) \in N \Leftarrow X \in N)$

$+1 \rightarrow$ s()

natural0(0).
natural0(s(X)) :- natural(X).



Příklad: Sčítání dvou čísel

$\mathrm{X} + \mathrm{Y} = \mathrm{Z}$

$\forall X : X + 0 = X$
$\forall X \forall Y \forall Z : X + (Y + 1) = (Z + 1) \Leftarrow X + Y = Z$

Peanova aritmetika: $1 = \mathrm{s}(0)$, $2 = \mathrm{s}(\mathrm{s}(0))$, $\mathrm{s}(\mathrm{Y})$ odpovídá $\mathrm{Y} + 1$

```
plus1(X,0,X).
plus1(X,s(Y),s(Z)) :- plus1(X,Y,Z).
```

Intensionální logiky

Úvodem. Módy pravdy

Cavaco Silva je presidentem Portugalska.

Jana čte.

Sluneční soustava má devět planet. (Nebo osm?)

Třetí odmocnina z 27 jsou 3.
nutně pravda, i v budoucnu. Ale:
Jana ví, že třetí odmocnina z 27 jsou 3.
Jan nevěří, že třetí odmocnina z 27 jsou 3.

Verifikace programů: nejen různé světy, ale i různé budoucnosti

VŽDY pravda, NĚKDY pravda, VÍM že, VĚŘÍM že, ...

Modální logika

$\Box\Phi$ - "nutně platí ϕ", "ϕ je vždy pravda"

$\Diamond\phi$ - "možná platí ϕ", "ϕ je někdy pravda

Je-li $\mathcal{L}$ jazyk predikátové logiky, rozšíříme ho na modální jazyk $\mathcal{L}_{\Box,\Diamond}$ přidáním dvou symbolů $\Box$ and $\Diamond$ do abecedy a do definice syntaxe přidáme

Je-li ϕ formule, pak také $(\Box\phi)$ a $(\Diamond\phi)$ jsou formule.

Sémantika: Kripkeho rámce