Resoluční metoda v predikátové logice

11. a 12. přednáška z LGR

Obsah

1. **Resoluční metoda**
2. **Převedení na klausální tvar**
   - Klausule v predikátové logice
   - Skolemizace
   - Převedení na klausální tvar
3. **Vytváření resolvent**
   - Unifikační algoritmus
   - Resolventy a resoluční princip

Resoluční metoda v predikátové logice

Resoluční metoda testuje, zda je množina klausulí splnitelná.

Je to universální metoda na testování sémantických problémů predikátové logiky, neboť

- mnohé problémy lze převést na problém (ne)splnitelnosti
- sentence lze převést na klausule se zachováním (ne)splnitelnosti

V predikátové logice bude resoluční metoda kopírovat to, jak funguje ve výrokové logice, ale obě fáze - převod na klausule i vytváření resolvent - budou komplikovanější.

Resoluční metoda v predikátové logice

Převedení problémů na problém (ne)splnitelnosti

- Sentence φ je kontradikce, právě když je nesplnitelná.
- Sentence φ je tautologie, právě když $\neg\varphi$ je nesplnitelná.
- Sémantický důsledek $S \models \varphi$ platí, právě když $S \cup \{\neg\varphi\}$ je nesplnitelná.
- Tautologická ekvivalence $\varphi \models\!\!\models \psi$ platí, právě když jsou obě množiny $\{\varphi, \neg\psi\}$ a $\{\psi, \neg\varphi\}$ nesplnitelné.

Převedení na klausální tvar

Definice

Klausule v predikátové logice je sentence, která má pouze obecné kvantifikátory, všechny jsou vpředu a za nimi následuje literál nebo disjunkce literálů. Prázdná klausule je ff.
Literál je atomická formule nebo negace atomické formule.

Příklad

V následujícím textu budeme používat jazyk s konstantním symbolem a, predikátovými symboly P, Q arity 1 a R arity 2; x, y, z budou proměnné. Formule níže jsou klausule:

$\forall x \forall y\,(P(x) \vee \neg R(a, y))$; $Q(a)$; ff

Převedení na klausální tvar

Převedení sentence na klausální tvar

1) Ve výrokové logice se formule upravila do CNF a rozdělila na klausule v místech konjunkcí, přitom veškeré úpravy zachovávaly tautologickou ekvivalenci.

2) V predikátové logice se sentence převede v podstatě také do CNF, ale vznikají tu problémy s kvantifikátory:

- všechny kvantifikátory musí být vpředu - tento problém jsme už minule vyřešili, umíme převést formuli do prenexního tvaru se zachováním tautologické ekvivalence;
- jsou dovoleny pouze obecné kvantifikátory - existenční kvantifikátory lze odstranit pomocí skolemizace, ta nezachová tautologickou ekvivalenci, nicméně zachová stejnou odpověď na otázku po splnitelnosti;

Převedení na klausální tvar

Definice

Množiny sentencí S a T jsou *ekvisplnitelné*, pokud jsou buď obě splnitelné, nebo obě nesplnitelné (aneb S má model, právě když T má model). Označíme $S \approx T$.

Zatímco ve výrokové logice jsme uměli pro S najít "tautologicky ekvivalentní" množinu klausulí $Kl(S)$, tj. obě množiny byly pravdivé ve stejných ohodnoceních, v predikátové logice bude klausální tvar $Kl(S)$ pouze ekvisplnitelný s množinou S, tj. S má model, právě když $Kl(S)$ má model, ale modely nemusí být stejné.

Převedení na klausální tvar

Odstranění existenčních kvantifikátorů se zachováním ekvisplnitelnosti řeší tzv. skolemizace.

Příklad 1.

Sentence $\exists x\, P(x)$ je ekvisplnitelná se sentencí $P(c)$, kde c je nový konstantní symbol.

- $P(c) \models \exists x\, P(x)$
- Z každé interpretace původního jazyka, která je modelem pro $\exists x\, P(x)$, lze vhodným interpretováním nového konstantního symbolu c vyrobit model pro $P(c)$.

Převedení na klausální tvar

Příklad 2.

Sentence $\forall x \exists y\, R(x, y)$ je ekvisplnitelná se sentencí $\forall x\, R(x, f(x))$, kde f je nový funkční symbol arity 1.

- $\forall x\, R(x, f(x)) \models \forall x \exists y\, R(x, y)$
- Z každé interpretace původního jazyka, která je modelem pro $\forall x \exists y\, R(x, y)$, lze vhodným interpretováním nového funkčního symbolu f vyrobit model pro $\forall x\, R(x, f(x))$.

Poznámka: Pro nekonečné universum U potřebujeme k vhodnému interpretování funkčního symbolu f axiom výběru.

Převedení na klausální tvar

Tvrzení

Nechť proměnná y je volná a jen volná ve formuli α. Označme $\alpha[y := t]$ formuli, která vznikne z formule α tak, že každý výskyt proměnné y nahradíme termem t.

- Sentence $\varphi = \exists y\, \alpha$ je ekvisplnitelná se sentencí $\varphi_s = \alpha[y := c]$, kde c je nový konstantní symbol (tzv. skolemizační konstantní symbol).
- Sentence $\varphi = \forall x_1 \ldots \forall x_n \exists y\, \alpha$ je ekvisplnitelná se sentencí $\varphi_s = \forall x_1 \ldots \forall x_n\, \alpha[y := f(x_1, \ldots, x_n)]$, kde f je nový funkční symbol arity n (tzv. skolemizační funkční symbol).

Převedení na klausální tvar

Tvrzení-pokračování

Pokud sentence φ obsahuje více existenčních kvantifikátorů, skolemizujeme je postupně od prvních (nejblíže ke kořeni synt. stromu) k posledním. Zavádíme stále nové skolemizační symboly, díky tomu výsledná φ_s bude ekvisplnitelná s φ. Speciálně platí:

- $\varphi_s \models \varphi$
- Z každé interpretace původního jazyka, která je modelem pro φ, lze vhodným interpretováním nových konstantních, resp. funkčních symbolů vyrobit model pro φ_s.

Poznámka

Skolemizace obohazuje původní jazyk, v němž byly sentence napsány.

Převedení na klausální tvar

Algoritmus 1.

K sentenci φ nalezneme ekvisplnitelnou množinu klausulí takto:

1. Přejmenujeme vázané proměnné tak, aby každý kvantifikátor vázal jinou proměnnou. *(To může obohatit jazyk o nové symboly pro proměnné.)*
2. Přepíšeme tautologicky ekvivalentně ostatní spojky na spojky $\neg$, $\wedge$, $\vee$, což lze, neboť tyto spojky tvoří úplný systém spojek.
3. Dostaneme negaci až před atomické podformule pomocí DeMorganových zákovů a zákonů pro negování kvantifikátorů.
4. Vytkneme kvantifikátory před konjunkce a disjunkce (což zachová tautologickou ekvivalenci, neboť proměnné vázané v různých podformulích se jmenují jinak).

Převedení na klausální tvar

5. Upravíme otevřenou podformuli za kvantifikátory do CNF - zbývá dostat $\wedge$ nad $\vee$ pomocí distributivního zákona.
6. Provedeme skolemizaci, čímž odstraníme existenční kvantifikátory (což zachová pouze ekvisplnitelnost). *Toto obohatí jazyk o nové konstantní či funkční symboly.*
7. Distribujeme obecné kvantifikátory pod konjunkci k těm podformulím, ve kterých se dané proměnné vyskytují (což zachová tautologickou ekvivalenci).
8. Rozdělíme sentenci v místech konjunkcí a vzniklé klausule tvoří $Kl(\varphi)$.

Úpravy vždy provádíme od kořene syntaktického stromu dané formule směrem k listům.

Převedení na klausální tvar

Tvrzení

Množina klausulí $Kl(\varphi)$ vytvořená podle předchozího algoritmu je ekvisplnitelná se sentencí φ.

Všechny kroky zachovali tautologickou ekvivalenci, kromě skolemizace. Ta ale zachová ekvisplnitelnost.

Tvrzení

Nechť S je konečná množina sentencí a nechť $Kl(S)$ je množina všech klausulí, které získáme postupným aplikováním předchozího algoritmu na jednotlivé sentence (přičemž u skolemizace zavádíme pro každou sentenci jiné nové symboly). Pak jsou množiny S a $Kl(S)$ ekvisplnitelné.

Převedení na klausální tvar

Poznámka

Pozor, nelze prohodit pořadí skolemizace a negace! Negování otočí kvantifikátory, takže skolemizovat pak budeme jiné proměnné. Speciálně při ověřování, zda $S \models \varphi$, zjišťujeme, zda $S \cup \{\neg\varphi\}$ je nesplnitelná a hledáme klausální tvar pro sentenci $\neg\varphi$!

Poznámka

Prenexní tvar sentence není určen jednoznačně, tudíž ani její klausální tvar - množina $Kl(S)$ není určena jednoznačně. Při vytýkání kvantifikátorů se hodí, aby existenční kvantifikátory byly co nejblíže ke kořeni syntaktického stromu, neboť bude jednodušší skolemizace (skolemizační funkční symbol bude menší arity).

Převedení na klausální tvar

Algoritmus 2.

Při převádění sentence na klausule není nutno jít přes prenexní tvar. Například po provedení kroků 1.-3. můžeme začít skolemizovat "po větvích syntaktického stromu": najdeme první existenční kvantifikátor ve větvi, proměnnou za ním skolemizujeme funkčním symbolem arity $n =$ počet obecných kvantifikátorů nad ním směrem ke kořeni, do něhož dosadíme proměnné, které jsou za těmito obecnými kvantifikátory.

Tento fakt je základem algoritmu č.2, jež je uveden ve skriptech Doc.Velebila.

Převedení na klausální tvar

Příklad

Jazyk L má konstantní symbol a, predikátové symboly P, Q arity 1 a R arity 2; x, y, z jsou proměnné.

Sentence $\varphi = [\forall x\, P(x) \Rightarrow \forall x \exists y\, (R(x,y) \wedge Q(a))]$
má prenexní tvar s otevřenou podformulí v CNF:
$\varphi_p = \exists x \forall z \exists y\, [(\neg P(x) \vee R(z,y)) \wedge (\neg P(x) \vee Q(a))]$

Skolemizace nahradí $x := c$, kde c je nový konstantní symbol,
$y := f(z)$, kde f je nový funkční symbol arity 1.
Skolemizovaná sentence má tvar:
$\varphi_s = \forall z\, [(\neg P(c) \vee R(z, f(z))) \wedge (\neg P(c) \vee Q(a))]$
Přitom $\varphi \nmid\!\models \varphi_p \approx \varphi_s$.

Klausální tvar $Kl(\varphi) = \{\forall z\, (\neg P(c) \vee R(z, f(z))); \neg P(c) \vee Q(a)\}$
obsahuje dvě klausule zapsané v obohaceném jazyce L'.

Vytváření resolvent

Resoluční metoda hledá odpověď na otázku, zda je množina klausulí S splnitelná, a to tak, že vytváří resolventy.

Resolventa dvojice klausulí α, β je klasule, která je sémantickým důsledkem množiny $\{\alpha, \beta\}$. Resoluční princip se opírá o fakt, že $S \models \mathrm{ff}$, právě když je S nesplnitelná. Vznikne-li při vytváření resolvent prázdná klausule ff, tehdy (a jen tehdy) je původní množina S nesplnitelná.

Vytváření resolvent

1) Ve výrokové logice existovala $\mathrm{res}(\alpha, \beta)$, pokud klausule obsahovaly komplementární literál (např. x, $\neg x$), a byla definována jako disjunkce všech ostatních literálů obsažených v α, β. Pravdivost x, $\neg x$ se vylučovala, proto v ohodnocení, ve kterém je pravdivá $\{\alpha, \beta\}$, musel být pravdivý jeden nebo druhý konec formulí α, β.

2) V predikátové logice se resolventa bude vyrábět podobně, ale potíž je v tom, že konflikt nemusí být vidět na první pohled. Komplementární literály nemusí být přímo v klausulích α, β, nýbrž v některých jejich sémantických důsledcích α', β'.

Vytváření resolvent

Příklady

Budeme používat jazyk s konstantními symboly a, b, funkčním symbolem f arity 1, predikátovými symboly P, Q, S arity 1, R arity 2 a T arity 3; x, y, z budou proměnné.

1. $\alpha = \forall x\, (P(x) \vee Q(x))$, $\beta = \forall y\, (\neg P(y) \vee S(a))$
   Vázanou proměnnou lze přejmenovat, $\beta = \forall x\, (\neg P(x) \vee S(a))$,
   konflikt nastává na celém universu U, tudíž
   $\{\alpha, \beta\} \models \forall x\, (Q(x) \vee S(a))$.

Vytváření resolvent

Příklady

2. $\alpha = \forall x\,(P(x) \vee Q(x))$, $\beta = \neg P(a) \vee S(a)$
   Pro $\alpha' = P(a) \vee Q(a)$ zřejmě platí $\alpha \models \alpha'$, konflikt nastává na prvku $[\![a]\!] \in U$, tudíž $\{\alpha, \beta\} \models Q(a) \vee S(a)$.
3. $\alpha = \forall x\,(P(x) \vee Q(x))$, $\beta = \forall y\,(\neg P(f(y)) \vee S(a))$
   Pro $\alpha' = \forall y\,(P(f(y)) \vee Q(f(y)))$ zřejmě platí $\alpha \models \alpha'$, konflikt nastává na podmnožině $Im([\![f]\!])$ všech výsledků zobrazení $[\![f]\!]$, tudíž $\{\alpha, \beta\} \models \forall y\,(Q(f(y)) \vee S(a))$.

Vytváření resolvent

Příklady

4. $\alpha = \forall x\,(P(f(x)) \vee Q(x))$, $\beta = \neg P(a) \vee S(a)$
   Nyní $\forall x\,P(f(x)) \not\models P(a)$ (ani $P(a) \not\models \forall x\,P(f(x))$), konflikt zde nenastává, pravdivost $\{\alpha, \beta\}$ lze zajistit vhodnou interpretací predikátu P, žádná resolventa nevzniká.
   Zde je model pro $\{\forall x\,P(f(x)), \neg P(a)\}$: $U = \mathbb{N}$, $[\![P]\!] = \{0\}$, $[\![f]\!] : \mathbb{N} \to \mathbb{N} : n \mapsto 0$, $[\![a]\!] = 1$.

Vytváření resolvent

Příklady

5. $\alpha = \forall x\,(R(x,x) \vee Q(x))$, $\beta = \forall y\,(\neg R(y, f(y)) \vee S(a))$
   Lze přejmenovat vázanou proměnnou,
   $\beta = \forall x\,(\neg R(x, f(x)) \vee S(a))$, ale to nepomůže, neboť
   $\forall x\,R(x,x) \not\models \forall x\,R(x, f(x))$ (ani naopak), konflikt zde nenastává, pravdivost $\{\alpha, \beta\}$ lze zajistit vhodnou interpretací predikátu R, žádná resolventa nevzniká.
   Zde je model pro $\{\forall x\,R(x,x), \forall y\,\neg R(y, f(y))\}$: $U = \mathbb{N}$, $[\![R]\!] = \{(m,n) \in \mathbb{N}^2, m = n\}$, $[\![f]\!] : \mathbb{N} \to \mathbb{N} : n \mapsto n+1$.

Vytváření resolvent

Povolené substituce

Označme $\alpha(x := t)$ klausuli, která vznikne substitucí termu t za všechny výskyty proměnné x v otevřeném jádru klausule α a následným doplněním (dopředu) obecných kvantifikátorů pro všechny proměnné, které v otevřeném jádru zůstaly.

Je-li $\vartheta = (x_1 := t_1, \ldots, x_n := t_n)$ seznam substitucí, pak $\alpha(\vartheta)$ je klausule, která vznikne z klausule α postupným aplikováním první až n–té substituce podle předchozího návodu.

Vytváření resolvent

Povolené substituce

Pozorování z předchozích příkladů se dají zobecnit:

- $\alpha \models \alpha(x := y)$, kde x lze legálně přejmenovat na y
- $\alpha \models \alpha(x := a)$, kde a je konstantní symbol
- $\alpha \models \alpha(x := f(y))$, kde f je funkční symbol arity 1
- $\alpha \models \alpha(x := g(t_1, \ldots, t_n))$, kde g je funkční symbol arity n a termy $t_1, \ldots, t_n$ neobsahují proměnnou x

Uvedené typy substitucí jsou povolenými substitucemi v resoluční metodě. Ostatní typy substitucí jsou zakázané, neboť nevedou k sémantickému důsledku.

Vytváření resolvent

Vraťme se k otázce, kdy z dvojice klausulí α, β vzniká resolventa.

Na příkladech jsme viděli, že pokud α obsahuje literál $P(t_1, \ldots, t_n)$ a β obsahuje literál $\neg P(s_1, \ldots, s_n)$, kde P je v tuto chvíli $n-$ární predikátový symbol a $t_1, \ldots, t_n$, $s_1, \ldots, s_n$ jsou termy, pak resolventa může, ale nemusí vzniknout. Záleží na tom, zda lze povolenými substitucemi lze vytvořit klausule α', β', které budou obsahovat komplementární literály $P(r_1, \ldots, r_n)$ a $\neg P(r_1, \ldots, r_n)$ (pro termy $r_1, \ldots, r_n$).
Aplikace pouze povolených substitucí zajistí, že $\alpha \models \alpha'$, $\beta \models \beta'$, tudíž $\{\alpha, \beta\} \models \mathrm{res}(\alpha', \beta')$, kde resolventu vytvoříme analogicky jako ve výrokové logice.

Vytváření resolvent

Unifikační algoritmus

Potřebujeme unifikovat dvě atomické formule (začínající stejným predikátovým symbolem), tj. potřebujeme je povolenými substitucemi převést na stejné formule. Navíc chceme dělat jen nezbytně nutné substituce, což v rámci vytváření resolvent znamená, že vytvoříme nejobecnější resolventu dvojice klausulí α, β (podchytíme konflikt v celé šíři).

Tento problém řeší *unifikační algoritmus*, který hledá tzv. maximální unifikátor, tj. seznam nezbytně nutných substitucí, které převedou dvě atomické formule na stejné, pokud to jde (= pokud se zachová sémantický důsledek).

Vytváření resolvent

Unifikační algoritmus

Vstup: Dvě (pozitivní) atomické formule L_1, L_2, které mají *různé proměnné*! (Kvůli zakázané substituci typu $x := f(x)$, nebo kvůli nebezpečí nelegálního přejmenování proměnných.)

Výstup: Maximální unifikátor ϑ, anebo hláška, že unifikátor neexistuje.

Hlavní myšlenka algoritmu:
Čteme řetězce L_1, L_2 zleva doprava, pokud jsou znaky v obou řetězcích stejné, tak je umažeme.
Pokud ne, pak se podíváme, zda je aspoň jeden z nich proměnná:
když ne, tak unifikátor neexistuje a skončíme;
pokud ano, ale v druhém řetězci je funkční symbol, v jehož argumentech se vyskytuje tatáž proměnná, tak unifikátor také neexistuje a skončíme.

Vytváření resolvent

Unifikační algoritmus

Nyní jsme v situaci, že znaky jsou různé a aspoň jeden z nich je proměnná a v druhém řetězci je proměnná, či konstantní symbol, či funkční symbol, v jehož argumentech se nevyskytuje tatáž proměnná. Můžeme tedy za proměnnou substituovat term z druhého řetězce. Tehdy přidáme do unifikátoru ϑ příslušnou povolenou substituci a zároveň ji provedeme na zbytky obou řetězců. Pak pokračujeme dále ve čtení a umazávání stejných znaků.
V případě, že se nám opakováním těchto kroků podaří dočíst řetězce L_1, L_2 až do konce a smazat je celé, tak máme v ϑ všechny potřebné a nutné substituce, ϑ je maximální unifikátor.

Podrobně je algoritmus sepsán ve skriptech doc. Velebila.

Vytváření resolvent

Příklad

- Maximální unifikátor pro $L_1 = T(x, y, x)$, $L_2 = T(a, f(z), z)$ je $\vartheta = (x := a, y := f(z), z := a)$.
  Unifikované řetězce mají tvar $L_1(\vartheta) = L_2(\vartheta) = T(a, f(a), a)$.
- Pro $L_1 = S(f(x))$, $L_2 = S(f(a))$, je $\vartheta = (x := a)$.
- Pro $L_1 = S(f(a))$, $L_2 = S(f(b))$, kde a, b jsou konstantní symboly, maximální unifikátor neexistuje.

Vytváření resolvent

Definice

Nechť α, β jsou klausule s *různými proměnnými* (žádná proměnná z α se nevyskytuje v β a naopak). Nechť dále α obsahuje literál $P(t_1, \ldots, t_n)$ a β obsahuje literál $\neg P(s_1, \ldots, s_n)$, kde P je $n-$ární predikátový symbol a $t_1, \ldots, t_n$, $s_1, \ldots, s_n$ jsou termy. A nechť existuje maximální unifikátor ϑ řetězců $L_1 = P(t_1, \ldots, t_n)$ a $L_2 = P(s_1, \ldots, s_n)$, který najdeme unifikačním algoritmem.
Tehdy a jen tehdy je definována *resolventa* klausulí α, β takto:

Aplikujeme substituce z max. unifikátoru ϑ na obě klausule, získáme klausule $\alpha(\vartheta)$, $\beta(\vartheta)$ s komplementárními literály $P(r_1, \ldots, r_n)$ a $\neg P(r_1, \ldots, r_n)$. Resolventa $\mathrm{res}(\alpha, \beta)$ je klausule, jejíž otevřené jádro tvoří disjunkce všech ostatních literálů z $\alpha(\vartheta)$, $\beta(\vartheta)$ (každý bereme jen jednou), vpředu jsou obecné kvantifikátory pro všechny proměnné, které se v otevřeném jádru vyskytují.

Vytváření resolvent

Příklad

- $\alpha = \forall x \forall y\, (\neg T(x, y, x) \vee S(y))$, $\beta = \forall z\, T(a, f(z), z)$
  Maximální unifikátor pro $L_1 = T(x, y, x)$, $L_2 = T(a, f(z), z)$ je $\vartheta = (x := a, y := f(z), z := a)$.
  Klausule po substituci jsou $\alpha(\vartheta) = \neg T(a, f(a), a) \vee S(f(a))$, $\beta(\vartheta) = T(a, f(a), a)$, resolventa $\mathrm{res}(\alpha, \beta) = S(f(a))$.
- $\alpha = \forall x \neg S(f(x))$, $\beta = S(f(a))$
  Maximální unifikátor pro $L_1 = S(f(x))$, $L_2 = S(f(a))$ je $\vartheta = (x := a)$. Klausule po substituci $\alpha(\vartheta) = \neg S(f(a))$, $\beta(\vartheta) = S(f(a))$, $\mathrm{res}(\alpha, \beta) = \mathrm{ff}$ (prázdná resolventa).
- $\alpha = \forall x \neg S(f(x))$, $\beta = S(a)$
  Maximální unifikátor pro $L_1 = S(f(x))$, $L_2 = S(a)$ neexistuje, resolventa $\mathrm{res}(\alpha, \beta)$ není definována.

Vytváření resolvent

Tvrzení

Nechť α, β jsou klausule s *různými proměnnými*.
Pak $\{\alpha, \beta\} \models \mathrm{res}(\alpha, \beta)$.

Maximální unifikátor ϑ jsme nalezli tak, aby $\alpha \models \alpha(\vartheta)$, $\beta \models \beta(\vartheta)$.
Dále podobně jako ve výrokové logice platí
$\{\alpha(\vartheta), \beta(\vartheta)\} \models \mathrm{res}(\alpha(\vartheta), \beta(\vartheta)) = \mathrm{res}(\alpha, \beta)$.
Díky transitivnosti sémantického důsledku $\{\alpha, \beta\} \models \mathrm{res}(\alpha, \beta)$.

Důsledek

Množiny $\{\alpha, \beta\}$ a $\{\alpha, \beta, \mathrm{res}(\alpha, \beta)\}$ mají stejné modely.

Vytváření resolvent

Definice

Nechť S je množina klauzulí s *různými proměnnými*. Označme

$$R(S) \;=\; S \cup \{\text{všechny resolventy dvojic z S}\}.$$

Položme

$$\begin{aligned} R^0(S) &= S \\ R^{i+1}(S) &= R(R^i(S)) \quad \text{pro} \quad i \in \mathbb{N} \\ R^\star(S) &= \bigcup\{R^i(S) \mid i \geq 0\}. \end{aligned}$$

Věta (Resoluční princip)

Konečná množina klauzulí S je nesplnitelná, právě když $R^\star(S)$ obsahuje $\square$.

Zpětná implikace plyne z faktu, že S a $R^\star(S)$ mají stejné modely.

Vytváření resolvent

Příklad

Je splnitelná množina klausulí S?
$S = \{\alpha = \forall x \forall y\,(\neg T(x, y, x) \vee S(y)); \beta = \forall z\; T(a, f(z), z);$
$\gamma = \forall t\,(R(b, t) \vee \neg S(t)); \delta = \forall v \forall w\, \neg R(v, f(w))\}$
Pozn.: Znaky t, v, w pro proměnné jsme museli přidat (k doposud používaným x, y, z), aby každá klausule mohla mít jiné proměnné.

- $\rho_1 = \mathrm{res}(\alpha, \beta) = S(f(a))$, maximální unifikátor je $\vartheta = (x := a, y := f(z), z := a)$.
- $\rho_2 = \mathrm{res}(\gamma, \delta) = \forall w\, \neg S(f(w))$, maximální unifikátor je $\vartheta = (v := b, t := f(w))$.
- $\mathrm{res}(\rho_1, \rho_2) = \square$, maximální unifikátor je $\vartheta = (w := a)$.

Podle resolučního principu je S nesplnitelná množina.

Vytváření resolvent

Poznámka

Resolventu jsme definovali jako ten nejobecnější sémantický důsledek dvojice klausulí. Pokud bychom použili substituce, které nejsou nezbytně nutné, nepopsali bychom konflikt v celé šíři a mohli bychom minout spor $\square$.

Viz předchozí příklad chybně:

- $\gamma = \forall t\,(R(b, t) \vee \neg S(t)); \delta = \forall v \forall w\, \neg R(v, f(w))$
  Příliš agresivní unifikátor pro $L_1 = R(b, t)$, $L_2 = R(v, f(w))$ je $\theta = (v := b, w := b, t := f(b))$.
  $\mathrm{res}(\gamma(\theta), \delta(\theta)) = \neg S(f(b))$, ovšem $\mathrm{res}(\gamma, \delta) \neq \neg S(f(b))$, i když platí $\{\gamma, \delta\} \models \neg S(f(b))$.
- $\mathrm{res}(\rho_1 = S(f(a)), \neg S(f(b)))$ neexistuje.

Vytváření resolvent

Poznámka

Proces vyrábění resolvent se v predikátové logice nemusí zastavit po konečném počtu kroků, přestože je množina klausulí S konečná.

Problém splnitelnosti množiny klausulí v predikátové logice je nerozhodnutelný problém = neexistuje universální algoritmus, který by pro libovolnou množinu klausulí zastavil a rozhodl správně o její splnitelnosti.
Tudíž i problém, zda je sentence tautologie je v preditátové logice nerozhodnutelný.

Vytváření resolvent

Příklad

Pro $S = \{\forall x\,(P(x) \vee \neg P(f(x)));\forall y\,(P(y) \vee \neg P(f(f(y))))\}$
se výroba resolvent nezastaví.
Maximální unifikátor bude vždy $\vartheta = (y := f(x))$ a vznikat budou resolventy $\mathrm{res}_n = \forall x\,(P(x) \vee \neg P(f^n(x))) = \forall y\,(P(y) \vee \neg P(f^n(y)))$ pro každé $n \geq 3$.
Přičemž $f^n(x) = f(f(\ldots f(x)\ldots))$, kde znak f je použit $n-$krát.

Resoluční metoda v predikátové logice

Příklad na závěr

Jazyk L má predikátový symbol R arity 2, x, y jsou proměnné.
Ověřte resoluční metodou:

1) $\varphi = \exists x \forall y\, R(x,y) \models \psi = \forall y \exists x\, R(x,y)$

- Zjistíme resoluční metodou, že $S = \{\varphi, \neg\psi\}$ je nesplnitelná.
- Převedeme do klausálního tvaru:
  $Kl(S) = \{\forall y\, R(a,y), \forall x\, \neg R(x,b)\}$, kde a, b jsou skolemizační konstantní symboly.
- Vyrábíme resolventy: maximální unifikátor
  $\vartheta = (x := a, y := b)$, resolventa $\rho = \mathrm{ff}$.
- Dle resolučního principu je $Kl(S)$ nesplnitelná a díky ekvisplnitelnosti je nesplnitelná také S.

Resoluční metoda v predikátové logice

Příklad na závěr

Jazyk L má predikátový symbol R arity 2, x, y jsou proměnné.
Ověřte resoluční metodou:

2) $\psi = \forall y \exists x\, R(x,y) \not\models \varphi = \exists x \forall y\, R(x,y)$

- Zjistíme resoluční metodou, že $S = \{\psi, \neg\varphi\}$ je splnitelná.
- Převedeme do klausálního tvaru:
  $Kl(S) = \{\forall y\, R(f(y),y), \forall x\, \neg R(x,g(x))\}$, kde f, g jsou skolemizační funkční symboly arity 1.
- Vyrábíme resolventy: maximální unifikátor neexistuje, žádná resolventa tedy neexistuje.
- Dle resolučního principu je $Kl(S)$ splnitelná a díky ekvisplnitelnosti je splnitelná také S.

Literatura

- J. Velebil: Velmi jemný úvod do matematické logiky. Kapitola 4.2. ftp://math.feld.cvut.cz/pub/velebil/y01mlo/logika.pdf
- M. Demlová, B. Pondělíček: Matematická logika, ČVUT Praha, 1997. Kapitola 14.