Kapitola 1

Rezoluce v predikátové logice

V této kapitole si ukážeme, jak lze adaptovat rezoluční metodu, kterou jsme představili v Kapitole **??**, na predikátovou logiku. Tato kapitola, poslední v části o predikátové logice, je poměrně rozsáhlá, proto uveďme přehled její struktury:

- Začneme neformálním úvodem (Sekce 1.1).

V následujících třech sekcích představíme nástroje, které nám umožní vypořádat se se specifiky predikátové logiky: s kvantifikátory, proměnnými a termy.

- V Sekci 1.2 si ukážeme si, jak pomocí *Skolemizace* odstranit kvantifikátory, abychom získali otevřené formule, které už lze převést do CNF.
- V Sekci 1.3 vysvětlíme, že rezoluční zamítnutí bychom mohli hledat 'na úrovni výrokové logiky' (tzv. *grounding*), pokud bychom nejprve za proměnné substituovali 'vhodné' konstantní termy.
- V Sekci 1.4 ukážeme, jak takové 'vhodné' substituce hledat pomocí *unifikačního algoritmu*.

Tím budeme mít všechny potřebné nástroje k představení vlastní rezoluční metody. Zbytek kapitoly má podobnou strukturu jako Kapitola **??**.

- Rezoluční pravidlo, rezoluční důkaz a související pojmy jsou popsány v Sekci 1.5.
- Sekce 1.6 je věnována důkazu korektnosti a úplnosti.
- Na závěr, v Sekci 1.7, popíšeme LI-rezoluci a její aplikaci v Prologu.

1.1 Úvod

Stejně jako ve výrokové logice, i v predikátové logice je rezoluční metoda založena na důkazu sporem. Chceme-li dokázat, že v teorii T platí sentence φ (tj. $T \models \varphi$), začneme s teorií $T \cup \{\neg\varphi\}$. Tuto teorii 'převedeme' do CNF, a výslednou množinu klauzulí S *zamítneme* rezolucí (tj. ukážeme, že $S \vdash_R \square$) čímž ukážeme, že je nesplnitelná.

Co myslíme konjunktivní normální formou? Roli *literálu* hraje *atomická formule*[1] nebo její negace. *Klauzule* je (v množinové reprezentaci) konečná množina literálů, a *formule* je množina

[1] Tj. $R(t_1, \ldots, t_n)$ resp. $t_1 = t_2$, kde t_i jsou L-termy a R je n-ární relační symbol z L.

klauzulí.[2] Jinak používáme stejnou terminologii, např. mluvíme o *pozitivních*, *negativních*, *opačných* literálech, $\square$ značí prázdnou klauzuli (která je nesplnitelná), apod.

Nejprve si neformálně ukážeme specifika rezoluce v predikátové logice na několika velmi jednoduchých příkladech.

Všimněme si nejprve, že jsou-li teorie T a sentence φ *otevřené* (neobsahují-li kvantifikátory), můžeme snadno sestrojit CNF formuli S *ekvivalentní* teorii $T \cup \{\neg\varphi\}$ (tj. mající stejnou množinu modelů). Nevadí ani univerzální kvantifikátory na začátku formule, ty můžeme odstranit beze změny významu.[3]

Příklad 1.1.1. Nechť $T = \{(\forall x)P(x), (\forall x)(P(x) \rightarrow Q(x))\}$ a $\varphi = (\exists x)Q(x)$. Je snadno vidět, že platí

$$T \sim \{P(x), P(x) \rightarrow Q(x)\} \sim \{P(x), \neg P(x) \vee Q(x)\}$$

a také:

$$\neg\varphi = \neg(\exists x)Q(x) \sim (\forall x)\neg Q(x) \sim \neg Q(x)$$

Teorii $T \cup \{\neg\varphi\}$ tedy můžeme převést na *ekvivalentní* CNF formuli

$$S = \{\{P(x)\}, \{\neg P(x), Q(x)\}, \{\neg Q(x)\}\}$$

kterou snadno zamítneme rezolucí ve dvou krocích. (Představte si místo $P(x)$ výrokovou proměnnou p a místo $Q(x)$ výrokovou proměnnou q.)

Obecně se nám to ale nepodaří, problémy dělá zejména existenční kvantifikátor. Na rozdíl od výrokové logiky *není* každá teorie ekvivalentní CNF formuli. Ukážeme si ale postup, kterým lze vždy najít *ekvisplnitelnou* CNF formuli, tj. takovou, která je nesplnitelná, *právě když* $T \cup \{\neg\varphi\}$ je nesplnitelná, což nám k důkazu sporem stačí. Této konstrukci se říká *Skolemizace* a spočívá v nahrazení existenčně kvantifikovaných proměnných nově přidanými konstantními resp. funkčními symboly.

Například, formuli $(\exists x)\psi(x)$ nahradíme formulí $\psi(x/c)$, kde c je nový konstantní symbol, který reprezentuje *svědka*, tj. prvek, díky kterému je existenční kvantifikátor splněn. Protože takových prvků může být mnoho, ztrácíme *ekvivalenci* teorií, platí ale, že je-li splnitelná původní formule, je splnitelná, i nová formule, a naopak.

Příklad 1.1.2. Máme-li $T = \{(\exists x)P(x), P(x) \leftrightarrow Q(x)\}$ a $\varphi = (\exists x)Q(x)$, potom

$$\neg\varphi \sim (\forall x)\neg Q(x) \sim \neg Q(x)$$

a ekvivalenci můžeme převést do CNF jako obvykle, dostáváme:

$$T \cup \{\neg\varphi\} \sim \{(\exists x)P(x), \neg P(x) \vee Q(x), \neg Q(x) \vee P(x), \neg Q(x)\}$$

Formuli $(\exists x)P(x)$ nyní nahradíme $P(c)$, kde c je nový konstantní symbol. Tím dostáváme CNF formuli:

$$S = \{\{P(c)\}, \{\neg P(x), Q(x)\}, \{\neg Q(x), P(x)\}, \{\neg Q(x)\}\}$$

Ta není ekvivalentní teorii $T \cup \{\neg\varphi\}$, ale je s ní *ekvisplnitelná* (v tomto případě jsou obě nesplnitelné).

[2] Jako ve výrokové logice připouštíme i nekonečné množiny klauzulí.

[3] Libovolná formule je ekvivalentní svému *generálnímu uzávěru*, a ekvivalence platí oběma směry.

Skolemizace může být i složitější, ne vždy stačí konstantní symbol. Pokud máme formuli tvaru $(\forall x)(\exists y)\psi(x,y)$, závisí zvolený svědek pro y na zvolené hodnotě pro x, tedy 'y je funkcí x'. V tomto případě musíme y nahradit $f(x)$, kde f je nový unární funkční symbol. Tím dostáváme formuli $(\forall x)\psi(x, y/f(x))$ a univerzální kvantifikátor nyní můžeme odstranit a psát jen $\psi(x, y/f(x))$, což už je otevřená formule, byť v jiném jazyce (rozšířeném o symbol f). Skolemizaci formálně popíšeme, a potřebné vlastnosti dokážeme, v Sekci 1.2.

Nyní se podívejme na *rezoluční pravidlo*. To je v predikátové logice složitější. Ukážeme si opět jen několik příkladů, formální definici necháme na později (Sekce 1.5).

Příklad 1.1.3. V předchozím příkladu jsme dospěli k následující CNF formuli S, která je nesplnitelná, a chtěli bychom ji tedy rezolucí zamítnout:

$$S = \{\{P(c)\}, \{\neg P(x), Q(x)\}, \{\neg Q(x), P(x)\}, \{\neg Q(x)\}\}$$

Pokud bychom se na ni podívali 'na úrovni výrokové logiky' ('ground level') a nahradili každou atomickou formuli novou výrokovou proměnnou, dostali bychom $\{\{r\}, \{\neg p, q\}, \{\neg q, p\}, \{\neg q\}\}$, což není nesplnitelné. Potřebujeme využít toho, že $P(c)$ a $P(x)$ mají 'podobnou strukturu' (jsou *unifikovatelné*).

Protože platí klauzule $\{\neg P(x), Q(x)\}$, platí i po provedení *libovolné substituce*, tj. klauzule $\{\neg P(x/t), Q(x/t)\}$ je důsledkem S pro libovolný term t. Mohli bychom si představit, že do S 'přidáváme' všechny takto získané klauzule.[4] Výsledná CNF formule by po převedení na 'úroveň výrokové logiky' už byla nesplnitelná.

Unifikační algoritmus nám ale rovnou řekne, že správná substituce je x/c, a toto zahrneme už do *rezolučního pravidla*, tedy *rezolventou* klauzulí $\{P(c)\}$ a $\{\neg P(x), Q(x)\}$ bude klauzule $\{Q(c)\}$.

Unifikace může být i složitější, a upozorněme ještě na jeden rozdíl oproti výrokové logice: dovolíme si udělat rezoluci přes více literálů najednou, a to v případě, že jsou všechny dohromady *unifikovatelné*:

Příklad 1.1.4. Z klauzulí $\{R(x, f(x)), R(g(y), z)\}$ a $\{\neg R(g(c), u), P(u)\}$ (kde R je binární relační, f a g jsou unární funkční, a c konstantní symbol) bude možné odvodit rezolventu $\{P(f(g(c))\}$ za použití *substituce* (*unifikace*) $\{x/g(c), y/c, z/f(g(c)), u/f(g(c))\}$, kde z první klauzule vybíráme *oba* literály najednou.

Poznámka 1.1.5. To, že proměnné mají 'lokální význam' v jednotlivých klauzulích (tj. můžeme za ně substituovat v jedné klauzuli aniž by to ovlivnilo ostatní klauzule), plyne z následující jednoduché tautologie, která platí pro libovolné formule ψ, χ (i pokud je v obou proměnná x volná):

$$\models (\forall x)(\psi \wedge \chi) \leftrightarrow (\forall x)\psi \wedge (\forall x)\chi$$

Jak je vidět v předchozím příkladě, budeme také vyžadovat, aby klauzule v rezolučním pravidle měly disjunktní množiny proměnných; toho lze dosáhnout přejmenováním proměnných, což je speciální případ substituce.

1.2 Skolemizace

V této sekci ukážeme postup, jak redukovat otázku splnitelnosti dané teorie T na otázku splnitelnosti *otevřené* teorie T'. Připomeňme, že T a T' obecně nebudou ekvivalentní, budou

[4] Těch je nekonečně mnoho, nekonečně mnoho je už jen *variant* jedné klauzule, tj. klauzulí vzniklých pouhým přejmenováním proměnných. To nám ale nevadí, CNF formule může být dle definice nekonečná.

ale *ekvisplnitelné*:

Definice 1.2.1 (Ekvisplnitelnost)**.** Mějme teorii T v jazyce L a teorii T' v ne nutně stejném jazyce L'. Říkáme, že T a T' jsou *ekvisplnitelné*, pokud platí:

$$T \text{ má model } \Leftrightarrow T' \text{ má model}$$

Celá konstrukce sestává z následujících kroků, které vysvětlíme níže:

1. Převod do *prenexní normální formy* (vytýkání kvantifikátorů).
2. Nahrazení formulí jejich generálními uzávěry (abychom získali sentence).
3. Odstranění existenčních kvantifikátorů (nahrazení sentencí *Skolemovými variantami*).
4. Odstranění zbývajících univerzálních kvantifikátorů (výsledkem jsou otevřené formule).

1.2.1 Prenexní normální forma

Nejprve ukážeme postup, jakým můžeme z libovolné formule 'vytknout' kvantifikátory, tj. převést do tzv. *prenexní normální formy*, která začíná posloupností kvantifikátorů, a pokračuje už jen volnou formulí.

Definice 1.2.2 (PNF)**.** Formule φ je v *prenexní normální formě (PNF)*, je-li tvaru

$$(Q_1 x_1) \dots (Q_n x_n) \varphi'$$

kde Q_i je kvantifikátor ($\forall$ nebo $\exists$) a formule φ' je otevřená. Formuli φ' potom říkáme *otevřené jádro* φ a $(Q_1 x_1) \dots (Q_n x_n)$ je *kvantifikátorový prefix.*

Je-li φ formule v PNF a jsou-li všechny kvantifikátory univerzální, potom říkáme, že φ je *univerzální* formule.

Cílem této podsekce je ukázat následující tvrzení:

Tvrzení 1.2.3 (Převod do PNF)**.** *Ke každé formuli φ existuje ekvivalentní formule v prenexní normální formě.*

Algoritmus bude podobně jako převod do CNF založen na nahrazování podformulí *ekvivalentními* podformulemi, s cílem posunout kvantifikátory blíže ke kořeni stromu formule. Co myslíme ekvivalencí formulí $\varphi \sim \varphi'$? To, že mají stejný význam, tj. v každém modelu a při každém ohodnocení proměnných mají touž pravdivostní hodnotu. Ekvivalentně, že platí $\models \varphi \leftrightarrow \varphi'$. Budeme potřebovat následující jednoduché pozorování:

Pozorování 1.2.4. *Nahradíme-li ve formuli φ nějakou podformuli ψ ekvivalentní formulí ψ', potom je i výsledná formule φ' ekvivalentní formuli φ.*

Převod je založen na opakovaném použití následujících syntaktických pravidel:

Lemma 1.2.5. *Označme jako $\overline{Q}$ kvantifikátor opačný ke Q. Nechť φ a ψ jsou formule, a proměnná x nechť není volná ve formuli ψ. Potom platí:*

$$\begin{aligned}
\neg(Qx)\varphi &\sim (\overline{Q}x)\neg\varphi \\
(Qx)\varphi \wedge \psi &\sim (Qx)(\varphi \wedge \psi) \\
(Qx)\varphi \vee \psi &\sim (Qx)(\varphi \vee \psi) \\
(Qx)\varphi \rightarrow \psi &\sim (\overline{Q}x)(\varphi \rightarrow \psi) \\
\psi \rightarrow (Qx)\varphi &\sim (Qx)(\psi \rightarrow \varphi)
\end{aligned}$$

Důkaz. Pravidla lze snadno ověřit sémanticky, nebo dokázat tablo metodou (v tom případě nejde-li o sentence, musíme je nahradit jejich generálními uzávěry). □

Všimněte si, že v pravidle $(Qx)\varphi{\to}\psi \sim (\overline{Q}x)(\varphi{\to}\psi)$ pro vytýkání z *antecendentu* implikace musíme změnit kvantifikátor (z $\forall$ na $\exists$ a naopak) zatímco při vytýkání z *konsekventu* zůstává kvantifikátor stejný. Proč tomu tak je vidíme nejlépe pokud přepíšeme implikaci pomocí disjunkce a negace:

$$(Qx)\varphi \to \psi \sim \neg(Qx)\varphi \vee \psi \sim (\overline{Q}x)(\neg\varphi) \vee \psi \sim (\overline{Q}x)(\neg\varphi \vee \psi) \sim (\overline{Q}x)(\varphi \to \psi)$$

Všimněte si také předpokladu, že x není volná v ψ. Bez něj by pravidla nefungovala, viz např:

$$(\exists x)P(x) \wedge Q(x) \ \not\sim\ (\exists x)(P(x) \wedge Q(x))$$

V takové situaci nahradíme formuli variantou, ve které přejmenujeme vázanou proměnnou x na nějakou novou proměnnou:

$$(\exists x)P(x) \wedge Q(x) \ \sim\ (\exists y)P(y) \wedge Q(x) \sim (\exists y)(P(y) \wedge Q(x))$$

Cvičení 1.1. Dokažte Pozorování 1.2.4 a všechna pravidla z Lemmatu 1.2.5.

Ukažme si postup na jednom příkladě:

Příklad 1.2.6. Převeďme formuli $((\forall z)P(x,z) \wedge P(y,z)) \ \to\ \neg(\exists x)P(x,y)$ do PNF. Zapíšeme jen jednotlivé mezikroky. Všimněte si, jaké pravidlo na jakou podformuli bylo použito (a také přejmenování proměnné v prvním kroku), a sledujte postup na stromu formule.

$$\begin{aligned}
&(\forall z)P(x,z) \wedge P(y,z) \ \to\ \neg(\exists x)P(x,y)\\
\sim\ &(\forall u)P(x,u) \wedge P(y,z) \ \to\ (\forall x)\neg P(x,y)\\
\sim\ &(\forall u)(P(x,u) \wedge P(y,z)) \ \to\ (\forall v)\neg P(v,y)\\
\sim\ &(\exists u)(P(x,u) \wedge P(y,z) \ \to\ (\forall v)\neg P(v,y))\\
\sim\ &(\exists u)(\forall v)(P(x,u) \wedge P(y,z) \ \to\ \neg P(v,y))
\end{aligned}$$

Nyní nám již nic nebrání dokázat Tvrzení 1.2.3:

Důkaz Tvrzení 1.2.3. Indukcí podle struktury formule φ s využitím Lemmatu 1.2.5 a Pozorování 1.2.4. □

Protože je každá formule $\varphi(x_1,\dots,x_n)$ ekvivalentní svému *generálnímu uzávěru*

$$(\forall x_1)\dots(\forall x_n)\varphi(x_1,\dots,x_n)$$

můžeme Tvrzení 1.2.3 vyslovit také takto:

Důsledek 1.2.7. *Ke každé formuli φ existuje ekvivalentní* sentence *v PNF.*

Například v Příkladu 1.2.6 je výsledná sentence $(\forall x)(\forall y)(\forall z)(\exists u)(\forall v)(P(x,u) \wedge P(y,z) \to \neg P(v,y))$.

Poznámka 1.2.8. Prenexní forma není jednoznačná, pravidla pro převod můžeme aplikovat v různém pořadí. Jak uvidíme v následující podsekci, je výhodné vytýkat přednostně kvantifikátory [ze kterých se stanou] existenční: Máme-li na výběr mezi $(\forall x)(\exists y)\varphi(x,y)$ a $(\exists y)(\forall x)\varphi(x,y)$, volíme druhou variantu, neboť v první je 'y závislé na x'.

1.2.2 Skolemova varianta

Nyní jsme převedli naše axiomy na ekvivalentní sentence v prenexním tvaru. Pokud by některá sentence obsahovala pouze univerzální kvantifikátory, tj. byla tvaru

$$(\forall x_1)\dots(\forall x_n)\varphi(x_1,\dots,x_n)$$

kde φ je otevřená, mohli bychom ji prostě nahradit jejím otevřeným jádrem φ, které je jí v tomto případě ekvivalentní. Jak si ale poradit s existenčními kvantifikátory, např. $(\exists x)\varphi(x)$, $(\forall x)(\exists y)\varphi(x,y)$, apod? Ty nejprve nahradíme jejich *Skolemovou variantou.*

Definice 1.2.9 (Skolemova varianta)**.** Mějme L-sentenci φ v PNF, a nechť všechny její vázané proměnné jsou různé. Nechť existenční kvantifikátory z prefixu φ jsou $(\exists y_1),\dots,(\exists y_n)$ (v tomto pořadí), a nechť pro každé i jsou $(\forall x_1),\dots,(\forall x_{n_i})$ právě všechny univerzální kvantifikátory předcházející kvantifikátor $(\exists y_i)$ v prefixu φ.

Označme L' rozšíření L o *nové* n_i-ární funkční symboly $f_1,\dots,f_n$, kde symbol f_i je arity n_i, pro každé i. *Skolemova varianta* sentence φ je L'-sentence φ_S vzniklá z φ tak, že pro každé $i=1,\dots,n$:

- odstraníme z prefixu kvantifikátor $(\exists y_i)$, a
- substituujeme za proměnnou y_i term $f_i(x_1,\dots,x_{n_i})$.

Tomuto procesu říkáme také *skolemizace.*

Příklad 1.2.10. Skolemova varianta sentence

$$\varphi=(\exists y_1)(\forall x_1)(\forall x_2)(\exists y_2)(\forall x_3)R(y_1,x_1,x_2,y_2,x_3)$$

je sentence

$$\varphi_S=(\forall x_1)(\forall x_2)(\forall x_3)R(f_1,x_1,x_2,f_2(x_1,x_2),x_3)$$

kde f_1 je nový konstantní symbol a f_2 je nový binární funkční symbol.

Poznámka 1.2.11. Nezapomeňte, že při skolemizaci musíme vycházet ze sentence! Máme-li formuli $(\exists y)E(x,y)$, není $E(x,c)$ její Skolemova varianta! Musíme napřed provést generální uzávěr $(\forall x)(\exists y)E(x,y)$, a potom správně skolemizovat jako $(\forall x)E(x,f(x))$, což je ekvivalentní otevřené formuli $E(x,f(x))$ (která říká něco mnohem slabšího než $E(x,c)$).

Je také důležité, aby každý symbol použitý při skolemizaci byl opravdu nový, jeho jedinou 'rolí' v celé teorii musí být reprezentovat 'existující' prvky v této formuli.

V následujícím lemmatu ukážeme klíčovou vlastnost skolemovy varianty:
[TODO]

Lemma 1.2.12.

Důkaz. □

Důsledek 1.2.13.

Vlastnosti Skolemovy varianty

Lemma *Nechť φ je sentence $(\forall x_1)\dots(\forall x_n)(\exists y)\psi$ jazyka L a φ' je sentence $(\forall x_1)\dots(\forall x_n)\psi(y/f(x_1,\dots,x_n))$, kde f je nový funkční symbol. Pak*

(1) *redukt $\mathcal{A}$ každého modelu $\mathcal{A}'$ formule φ' na jazyk L je modelem φ,*

(2) *každý model $\mathcal{A}$ formule φ lze expandovat na model $\mathcal{A}'$ formule φ'.*

Poznámka *Na rozdíl od extenze o definici funkčního symbolu, expanze v tvrzení* (2) *tentokrát nemusí být jednoznačná.*

Důkaz (1) Nechť $\mathcal{A}' \models \varphi'$ a $\mathcal{A}$ je redukt $\mathcal{A}'$ na jazyk L. Jelikož pro každé ohodnocení e je $\mathcal{A} \models \psi[e(y/a)]$, kde $a = (f(x_1,\dots,x_n))^{A'}[e]$, platí $\mathcal{A} \models \varphi$.
(2) Nechť $\mathcal{A} \models \varphi$. Pak existuje funkce $f^A\colon A^n \to A$ taková, že pro každé ohodnocení e platí $\mathcal{A} \models \psi[e(y/a)]$, kde $a = f^A(e(x_1),\dots,e(x_n))$, a tedy expanze $\mathcal{A}'$ struktury $\mathcal{A}$ o funkci f^A je modelem φ'. $\square$

Důsledek *Je-li φ' Skolemova varianta formule φ, obě tvrzení* (1) *a* (2) *pro φ, φ' rovněž platí. Tedy φ, φ' jsou ekvisplnitelné.*

1.2.3 Skolemova věta

[TODO]

Skolemova věta

Věta *Každá teorie T má otevřenou konzervativní extenzi T^*.*

Důkaz Lze předpokládat, že T je v uzavřeném tvaru. Nechť L je její jazyk.

- Nahrazením každého axiomu teorie T za ekvivalentní formuli v prenexním tvaru získáme ekvivalentní teorii T°.
- Nahrazením každého axiomu teorie T° za jeho Skolemovu variantu získáme teorii T' rozšířeného jazyka L'.
- Jelikož je redukt každého modelu teorie T' na jazyk L modelem teorie T, je T' extenze T.
- Jelikož i každý model teorie T lze expandovat na model teorie T', je to extenze konzervativní.
- Jelikož každý axiom teorie T' je univerzální sentence, jejich nahrazením za otevřená jádra získáme otevřenou teorii T^* ekvivalentní s T'. $\square$

Důsledek *Ke každé teorii existuje ekvisplnitelná otevřená teorie.*

1.3 Grounding

[TODO]

Redukce nesplnitelnosti na úroveň VL

Je-li otevřená teorie nesplnitelná, lze to "doložit na konkrétních prvcích".

Např. teorie
$$T = \{P(x,y) \vee R(x,y),\ \neg P(c,y),\ \neg R(x,f(x))\}$$
jazyka $L = \langle P, R, f, c\rangle$ nemá model, což lze doložit nesplnitelnou konjunkcí konečně mnoha instancí (některých) axiomů teorie T v konstantních termech
$$(P(c,f(c)) \vee R(c,f(c)))\ \wedge\ \neg P(c,f(c))\ \wedge\ \neg R(c,f(c)),$$
což je lživá formule ve tvaru výroku
$$(p \vee r)\ \wedge\ \neg p\ \wedge\ \neg r.$$

Instance $\varphi(x_1/t_1,\dots,x_n/t_n)$ otevřené formule φ ve volných proměnných $x_1,\dots,x_n$ je *základní (ground) instance*, jsou-li všechny termy $t_1,\dots,t_n$ konstantní. Konstantní termy nazýváme také *základní (ground) termy.*

Přímá redukce do VL

Herbrandova věta umožňuje následující postup. Je ale značně neefektivní.

- Nechť S je *(vstupní)* formule v množinové reprezentaci.
- Lze předpokládat, že jazyk obsahuje alespoň jeden konstantní symbol.
- Nechť S' je množina všech základních instancí klauzulí z S.
- Zavedením prvovýroků pro každou atomickou sentenci lze S' převést na (případně nekonečnou) výrokovou formuli v množinové reprezentaci.
- Rezolucí na úrovni VL ověříme její nesplnitelnost.

Např. pro $S = \{\{P(x,y), R(x,y)\}, \{\neg P(c,y)\}, \{\neg R(x,f(x))\}\}$ *je*
$$S' = \{\{P(c,c), R(c,c)\}, \{P(c,f(c)), R(c,f(c))\}, \{P(f(c),f(c)), R(f(c),f(c))\}\dots,$$
$$\{\neg P(c,c)\}, \{\neg P(c,f(c))\}, \dots, \{\neg R(c,f(c))\}, \{\neg R(f(c),f(f(c)))\}, \dots\}$$
nesplnitelná, neboť na úrovni VL je
$$S' \supseteq \{\{P(c,f(c)), R(c,f(c))\}, \{\neg P(c,f(c))\}, \{\neg R(c,f(c))\}\} \vdash_R \square.$$

1.3.1 Herbrandův model

[TODO]

Herbrandův model

Nechť $L = \langle \mathcal{R}, \mathcal{F} \rangle$ je jazyk s alespoň jedním konstantním symbolem.

(Je-li třeba, do L přidáme nový konstantní symbol.)

- *Herbrandovo univerzum* pro L je množina všech konstantních termů z L.

  Např. pro $L = \langle P, f, c \rangle$, kde P je relační, f je binární funkční, c konstantní
  $$A = \{c, f(c,c), f(f(c,c),c), f(c, f(c,c)), f(f(c,c), f(c,c)), \dots\}$$

- Struktura $\mathcal{A}$ pro L je *Herbrandova struktura*, je-li doména A Herbrandovo univerzum pro L a pro každý n-ární funkční symbol $f \in \mathcal{F}$ a $t_1, \dots, t_n \in A$,
  $$f^A(t_1, \dots, t_n) = f(t_1, \dots, t_n)$$
  (včetně $n = 0$, tj. $c^A = c$ pro každý konstantní symbol c).

  Poznámka Na rozdíl od kanonické struktury nejsou předepsané relace.

  Např. $\mathcal{A} = \langle A, P^A, f^A, c^A \rangle$, kde $P^A = \emptyset$, $c^A = c$ a $f^A(c,c) = f(c,c)$, ...

- *Herbrandův model* teorie T je Herbrandova struktura, jež je modelem T.

1.3.2 Herbrandova věta

[TODO]

Herbrandova věta

Věta *Nechť T je otevřená teorie jazyka L bez rovnosti a s alespoň jedním konstantním symbolem. Pak*

(a) *T má Herbrandův model, anebo*

(b) *existuje konečně mnoho základních instancí axiomů z T, jejichž konjunkce je nesplnitelná, a tedy T nemá model.*

Důkaz Nechť T' je množina všech základních instancí axiomů z T. Uvažme dokončené (např. systematické) tablo τ z T' v jazyce L (bez přidávání nových konstant) s položkou $F\bot$ v kořeni.

- Obsahuje-li tablo τ bezespornou větev V, kanonický model z větve V je Herbrandovým modelem teorie T.
- Jinak je τ sporné, tj. $T' \vdash \bot$. Navíc je konečné, tedy $\bot$ je dokazatelný jen z konečně mnoha formulí T', tj. jejich konjunkce je nesplnitelná. $\square$

Poznámka V případě jazyka L s rovností teorii T rozšíříme na T^* o axiomy rovnosti pro L a pokud T^* má Herbrandův model $\mathcal{A}$, zfaktorizujeme ho dle $=^A$.

1.3.3 Důsledky

[TODO]

Důsledky Herbrandovy věty

Nechť L je jazyk obsahující alespoň jeden konstantní symbol.

Důsledek *Pro každou otevřenou $\varphi(x_1, \ldots, x_n)$ jazyka L je $(\exists x_1) \ldots (\exists x_n)\varphi$ pravdivá, právě když existují konstantní termy t_{ij} jazyka L takové, že*

$$\varphi(x_1/t_{11}, \ldots, x_n/t_{1n}) \vee \ldots \vee \varphi(x_1/t_{m1}, \ldots, x_n/t_{mn})$$

je (výroková) tautologie.

Důkaz $(\exists x_1) \ldots (\exists x_n)\varphi$ je pravdivá $\Leftrightarrow$ $(\forall x_1) \ldots (\forall x_n)\neg\varphi$ je nesplnitelná $\Leftrightarrow$ $\neg\varphi$ je nesplnitelná. Ostatní vyplývá z Herbrandovy věty pro $\neg\varphi$. $\square$

Důsledek *Otevřená teorie T jazyka L má model, právě když teorie T' všech základních instancí axiomů z T má model.*

Důkaz Má-li T model $\mathcal{A}$, platí v něm každá instance každého axiomu z T, tedy $\mathcal{A}$ je modelem T'. Nemá-li T model, dle H. věty existuje (konečně) formulí z T', jejichž konjunkce je nesplnitelná, tedy T' nemá model. $\square$

1.4 Unifikace

[TODO]

1.4.1 Substituce

[TODO]

Substituce - příklady

Efektivnější je využívat vhodných substitucí. Např. pro

a) $\{P(x), Q(x,a)\}$, $\{\neg P(y), \neg Q(b,y)\}$ substitucí x/b, y/a dostaneme $\{P(b), Q(b,a)\}$, $\{\neg P(a), \neg Q(b,a)\}$ a z nich rezolucí $\{P(b), \neg P(a)\}$.

Nebo substitucí x/y a rezolucí dle $P(y)$ dostaneme $\{Q(y,a), \neg Q(b,y)\}$.

b) $\{P(x), Q(x,a), Q(b,y)\}$, $\{\neg P(v), \neg Q(u,v)\}$ substituce x/b, y/a, u/b, v/a dává $\{P(b), Q(b,a)\}$, $\{\neg P(a), \neg Q(b,a)\}$ a z nich rezolucí $\{P(b), \neg P(a)\}$.

c) $\{P(x), Q(x,z)\}$, $\{\neg P(y), \neg Q(f(y),y)\}$ substitucí $x/f(z)$, y/z dostaneme $\{P(f(z)), Q(f(z),z)\}$, $\{\neg P(z), \neg Q(f(z),z)\}$ a z nich $\{P(f(z)), \neg P(z)\}$.

Při substituci $x/f(a)$, y/a, z/a dostaneme $\{P(f(a)), Q(f(a),a)\}$, $\{\neg P(a), \neg Q(f(a),a)\}$ a z nich rezolucí $\{P(f(a)), \neg P(a)\}$. Předchozí substituce je ale obecnější.

Substituce

- *Substituce* je (konečná) množina $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$, kde x_i jsou navzájem různé proměnné a t_i jsou termy, přičemž t_i není x_i.
- Jsou-li všechny termy t_i konstantní, je σ *základní substituce.*
- Jsou-li t_i navzájem různé proměnné, je σ *přejmenování proměnných.*
- *Výraz* je literál nebo term. *(Substituci lze aplikovat na výrazy.)*
- *Instance* výrazu E *při substituci* $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ je výraz $E\sigma$ vzniklý z E současným nahrazením všech výskytů proměnných x_i za t_i.
- Pro množinu výrazů S označme $S\sigma$ množinu instancí $E\sigma$ výrazů E z S.

Poznámka Jelikož substituce je současná pro všechny proměnné zároveň, případný výskyt proměnné x_i v termu t_j nevede k zřetězení substitucí.
Např. pro $S = \{P(x), R(y,z)\}$ a substituci $\sigma = \{x/f(y,z), y/x, z/c\}$ je

$$S\sigma = \{P(f(y,z)), R(x,c)\}.$$

Skládání substitucí

Zadefinujeme $\sigma\tau$ tak, aby $E(\sigma\tau) = (E\sigma)\tau$ pro každý výraz E.

Např. pro $E = P(x,w,u)$, $\sigma = \{x/f(y), w/v\}$, $\tau = \{x/a, y/g(x), v/w, u/c\}$ je

$$E\sigma = P(f(y), v, u), \quad (E\sigma)\tau = P(f(g(x)), w, c).$$

Pak by mělo být $\sigma\tau = \{x/f(g(x)), y/g(x), v/w, u/c\}$.

Pro substituce $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ a $\tau = \{y_1/s_1, \ldots, y_n/s_n\}$ definujeme

$$sigma\tau = \{x_i/t_i\tau \mid x_i \in X,\ x_i \text{ není } t_i\tau\} \cup \{y_j/s_j \mid y_j \in Y \setminus X\}$$

složenou substituci σ a τ, kde $X = \{x_1, \ldots, x_n\}$ a $Y = \{y_1, \ldots, y_m\}$.

Poznámka Skládání substitucí není komutativní, např. pro uvedené σ a τ je

$$\tau\sigma = \{x/a, y/g(f(y)), u/c, w/v\} \neq \sigma\tau.$$

Skládání substitucí - vlastnosti

Ukážeme, že definice vyhovuje našemu požadavku a skládání je asociativní.

Tvrzení *Pro každý výraz E a substituce σ, τ, ϱ platí*

(i) $(E\sigma)\tau = E(\sigma\tau)$,

(ii) $(\sigma\tau)\varrho = \sigma(\tau\varrho)$.

Důkaz Nechť $\sigma = \{x_1/t_1, \dots, x_n/t_n\}$ a $\tau = \{y_1/s_1, \dots, y_m/s_m\}$. Stačí uvážit případ, kdy E je proměnná, řekněme v.

(i) Je-li v proměnná x_i pro nějaké i, je $v\sigma = t_i$ a $(v\sigma)\tau = t_i\tau$, což je $v(\sigma\tau)$ dle definice $\sigma\tau$. Jinak $v\sigma = v$ a $(v\sigma)\tau = v\tau$.

Je-li v proměnná y_j pro nějaké j, je dále $(v\sigma)\tau = v\tau = s_j$, což je $v(\sigma\tau)$ dle definice $\sigma\tau$. Jinak $(v\sigma)\tau = v\tau = v$ a zároveň $v(\sigma\tau) = v$.

(ii) Opakovaným užitím (i) dostaneme pro každý výraz E,

$$E((\sigma\tau)\varrho) = (E(\sigma\tau))\varrho = ((E\sigma)\tau)\varrho = (E\sigma)(\tau\varrho) = E(\sigma(\tau\varrho)). \qquad \square$$

1.4.2 Unifikační algoritmus

[TODO]

Unifikace

Nechť $S = \{E_1, \dots, E_n\}$ je (konečná) množina výrazů.

- *Unifikace* pro S je substituce σ taková, že $E_1\sigma = E_2\sigma = \dots = E_n\sigma$, tj. $S\sigma$ je singleton.
- S je *unifikovatelná*, pokud má unifikaci.
- Unifikace σ pro S je *nejobecnější unifikace (mgu)*, pokud pro každou unifikaci τ pro S existuje substituce λ taková, že $\tau = \sigma\lambda$.

Např. $S = \{P(f(x), y), P(f(a), w)\}$ je unifikovatelná pomocí nejobecnější unifikace $\sigma = \{x/a, y/w\}$. Unifikaci $\tau = \{x/a, y/b, w/b\}$ dostaneme jako $\sigma\lambda$ pro $\lambda = \{w/b\}$. τ není mgu, nelze z ní získat unifikaci $\varrho = \{x/a, y/c, w/c\}$.

Pozorování Jsou-li σ, τ různé nejobecnější unifikace pro S, liší se pouze přejmenováním proměnných.

Unifikační algoritmus

Nechť S je (konečná) neprázdná množina výrazů a p je nejlevější pozice, na které se nějaké dva výrazy z S liší. Pak *neshoda* v S je množina $D(S)$ podvýrazů začínajících na pozici p ze všech výrazů v S .

Např. pro $S = \{P(x, y), P(f(x), z), P(z, f(x))\}$ je $D(S) = \{x, f(x), z\}$.

Vstup Neprázdná (konečná) množina výrazů S.
Výstup Nejobecnější unifikace σ pro S nebo *"S není unifikovatelná"*.

(0) Nechť $S_0 := S$, $\sigma_0 := \emptyset$, $k := 0$. *(inicializace)*

(1) Je-li S_k singleton, vydej substituci $\sigma = \sigma_0\sigma_1 \cdots \sigma_k$. *(mgu pro S)*

(2) Zjisti, zda v $D(S_k)$ existuje proměnná x a term t neobsahující x.

(3) Pokud ne, vydej *"S není unifikovatelná"*.

(4) Jinak $\sigma_{k+1} := \{x/t\}$, $S_{k+1} := S_k\sigma_{k+1}$, $k := k+1$ a jdi na (1).

Poznámka Test výskytu proměnné x v termu t v kroku (2) *může být* "drahý".

Unifikační algoritmus - příklad

$$S = \{P(f(y, g(z)), h(b)),\ P(f(h(w), g(a)), t),\ P(f(h(b), g(z)), y)\}$$

1) $S_0 = S$ není singleton a $D(S_0) = \{y, h(w), h(b)\}$ obsahuje term $h(w)$ a proměnnou y nevyskytující se v $h(w)$. Pak $\sigma_1 = \{y/h(w)\}$, $S_1 = S_0\sigma_1$, tj.
$$S_1 = \{P(f(h(w), g(z)), h(b)),\ P(f(h(w), g(a)), t),\ P(f(h(b), g(z)), h(w))\}.$$

2) $D(S_1) = \{w, b\}$, $\sigma_2 = \{w/b\}$, $S_2 = S_1\sigma_2$, tj.
$$S_2 = \{P(f(h(b), g(z)), h(b)),\ P(f(h(b), g(a)), t)\}.$$

3) $D(S_2) = \{z, a\}$, $\sigma_3 = \{z/a\}$, $S_3 = S_2\sigma_3$, tj.
$$S_3 = \{P(f(h(b), g(a)), h(b)),\ P(f(h(b), g(a)), t)\}.$$

4) $D(S_3) = \{h(b), t\}$, $\sigma_4 = \{t/h(b)\}$, $S_4 = S_3\sigma_4$, tj.
$$S_4 = \{P(f(h(b), g(a)), h(b))\}.$$

5) S_4 je singleton a nejobecnější unifikace pro S je
$$\sigma = \{y/h(w)\}\{w/b\}\{z/a\}\{t/h(b)\} = \{y/h(b), w/b, z/a, t/h(b)\}.$$

Unifikační algoritmus - korektnost

Tvrzení *Pro každé S unifikační algoritmus vydá po konečně mnoha krocích korektní výsledek, tj. nejobecnější unifikaci σ pro S nebo pozná, že S není unifikovatelná. (∗) Navíc, pro každou unifikaci τ pro S platí, že $\tau = \sigma\tau$.*

Důkaz V každém kroku eliminuje jednu proměnnou, někdy tedy skončí.

- Skončí-li neúspěchem po k krocích, nelze unifikovat $D(S_k)$, tedy ani S.
- Vydá-li $\sigma = \sigma_0\sigma_1 \cdots \sigma_k$, je σ evidentně unifikace pro S.
- Dokážeme-li, že σ má vlastnost (∗), je σ nejobecnější unifikace pro S.

(1) Nechť τ je unifikace pro S. Ukážeme, že $\tau = \sigma_0\sigma_1 \cdots \sigma_i\tau$ pro každé $i \leq k$.

(2) Pro $i = 0$ platí (1). Nechť $\sigma_{i+1} = \{x/t\}$, předpokládejme $\tau = \sigma_0\sigma_1 \cdots \sigma_i\tau$.

(3) Stačí dokázat, že $v\sigma_{i+1}\tau = v\tau$ pro každou proměnnou v.

(4) Pro $v \neq x$ je $v\sigma_{i+1} = v$, tedy platí (3). Nyní $v = x$ a $v\sigma_{i+1} = x\sigma_{i+1} = t$.

(5) Jelikož τ unifikuje $S_i = S\sigma_0\sigma_1 \cdots \sigma_i$ a proměnná x i term t jsou v $D(S_i)$, musí τ unifikovat x a t, tj. $t\tau = x\tau$, jak bylo požadováno pro (3). $\square$

1.5 Rezoluční metoda

[TODO]

1.5.1 Rezoluční pravidlo

[TODO]

Nechť klauzule C_1, C_2 neobsahují stejnou proměnnou a jsou ve tvaru

$$C_1 = C_1' \sqcup \{A_1, \ldots, A_n\}, \quad C_2 = C_2' \sqcup \{\neg B_1, \ldots, \neg B_m\},$$

kde $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ lze unifikovat a $n, m \geq 1$. Pak klauzule

$$C = C_1'\sigma \cup C_2'\sigma,$$

kde σ je nejobecnější unifikace pro S, je *rezolventa* klauzulí C_1 a C_2.

Např. v klauzulích $\{P(x), Q(x,z)\}$ *a* $\{\neg P(y), \neg Q(f(y), y)\}$ *lze unifikovat* $S = \{Q(x,z), Q(f(y), y)\}$ pomocí nejobecnější unifikace $\sigma = \{x/f(y), z/y\}$ a získat z nich rezolventu $\{P(f(y)), \neg P(y)\}$.

Poznámka Podmínce o různých proměnných lze vyhovět přejmenováním proměnných v rámci klauzule. Je to nutné, např. z $\{\{P(x)\}, \{\neg P(f(x))\}\}$ *lze po přejmenování získat* $\square$*, ale* $\{P(x), P(f(x))\}$ *nelze unifikovat.*

1.5.2 Rezoluční důkaz

[TODO]

Rezoluční důkaz

Pojmy zavedeme jako ve VL, jen navíc dovolíme přejmenování proměnných.

- *Rezoluční důkaz (odvození)* klauzule C z formule S je konečná posloupnost $C_0, \ldots, C_n = C$ taková, že pro každé $i \leq n$ je $C_i = C_i'\sigma$, kde $C_i' \in S$ a σ je přejmenování proměnných, nebo je C_i rezolventou nějakých dvou předchozích klauzulí (i stejných).

- Klauzule C je (rezolucí) *dokazatelná* z S, psáno $S \vdash_R C$, pokud má rezoluční důkaz z S.
- *Zamítnutí* formule S je rezoluční důkaz $\square$ z S.
- S je (rezolucí) *zamítnutelná*, pokud $S \vdash_R \square$.

Poznámka Eliminace více literálů najednou je někdy nezbytná, např. $S = \{\{P(x), P(y)\}, \{\neg P(x), \neg P(y)\}\}$ *je rezolucí zamítnutelná, ale nemá zamítnutí, při kterém by se v každém kroku eliminoval pouze jeden literál.*

Příklad rezoluce

Mějme teorii $T = \{\neg P(x,x),\ P(x,y) \to P(y,x),\ P(x,y) \wedge P(y,z) \to P(x,z)\}$.

Je $T \models (\exists x)\neg P(x, f(x))$? Tedy, je následující formule T' nesplnitelná?

$$T' = \{\{\neg P(x,x)\}, \{\neg P(x,y), P(y,x)\}, \{\neg P(x,y), \neg P(y,z), P(x,z)\}, \{P(x, f(x))\}\}$$

`files/rezolucePLpriklad.pdf`

1.6 Korektnost a úplnost

[TODO]

1.6.1 Věta o korektnosti

[TODO]

Nejprve ukážeme, že obecné rezoluční pravidlo je korektní.

Tvrzení Nechť C je rezolventa klauzulí C_1, C_2. Pro každou L-strukturu $\mathcal{A}$,

$$\mathcal{A} \models C_1 \text{ a } \mathcal{A} \models C_2 \quad \Rightarrow \quad \mathcal{A} \models C.$$

Důkaz Nechť $C_1 = C_1' \sqcup \{A_1, \ldots, A_n\}$, $C_2 = C_2' \sqcup \{\neg B_1, \ldots, \neg B_m\}$, σ je nejobecnější unifikace pro $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ a $C = C_1'\sigma \cup C_2'\sigma$.

- Jelikož C_1, C_2 jsou otevřené, platí i $\mathcal{A} \models C_1\sigma$ a $\mathcal{A} \models C_2\sigma$.
- Máme $C_1\sigma = C_1'\sigma \cup \{S\sigma\}$ a $C_2\sigma = C_2'\sigma \cup \{\neg(S\sigma)\}$.

- Ukážeme, že $\mathcal{A} \models C[e]$ pro každé e. Je-li $\mathcal{A} \models S\sigma[e]$, pak $\mathcal{A} \models C_2'\sigma[e]$ a tedy $\mathcal{A} \models C[e]$. Jinak $\mathcal{A} \not\models S\sigma[e]$, pak $\mathcal{A} \models C_1'\sigma[e]$ a tedy $\mathcal{A} \models C[e]$. $\square$

Věta (korektnost) *Je-li formule S rezolucí zamítnutelná, je S nesplnitelná.*

Důkaz Nechť $S \vdash_R \square$. Kdyby $\mathcal{A} \models S$ pro nějakou strukturu $\mathcal{A}$, z korektnosti rezolučního pravidla by platilo i $\mathcal{A} \models \square$, což není možné. $\square$

1.6.2 Lifting lemma

[TODO]

Lifting lemma

Rezoluční důkaz na úrovni VL lze "zdvihnout" na úroveň PL.

Lemma *Nechť $C_1^* = C_1\tau_1$, $C_2^* = C_2\tau_2$ jsou základní instance klauzulí C_1, C_2 neobsahující stejnou proměnnou a C^* je rezolventa C_1^* a C_2^*. Pak existuje rezolventa C klauzulí C_1 a C_2 taková, že $C^* = C\tau_1\tau_2$ je základní instance C.*

Důkaz Předpokládejme, že C^* je rezolventa C_1^*, C_2^* přes literál $P(t_1, \ldots, t_k)$.

- Pak lze psát $C_1 = C_1' \sqcup \{A_1, \ldots, A_n\}$ a $C_2 = C_2' \sqcup \{\neg B_1, \ldots, \neg B_m\}$, kde $\{A_1, \ldots, A_n\}\tau_1 = \{P(t_1, \ldots, t_k)\}$ a $\{\neg B_1, \ldots, \neg B_m\}\tau_2 = \{\neg P(t_1, \ldots, t_k)\}$.
- Tedy $(\tau_1\tau_2)$ unifikuje $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ a je-li σ mgu pro S z unifikačního algoritmu, pak $C = C_1'\sigma \cup C_2'\sigma$ je rezolventa C_1 a C_2.
- Navíc $(\tau_1\tau_2) = \sigma(\tau_1\tau_2)$ z vlastnosti $(*)$ pro σ a tedy

$$\begin{aligned} C\tau_1\tau_2 &= (C_1'\sigma \cup C_2'\sigma)\tau_1\tau_2 = C_1'\sigma\tau_1\tau_2 \cup C_2'\sigma\tau_1\tau_2 = C_1'\tau_1 \cup C_2'\tau_2 \\ &= (C_1 \setminus \{A_1, \ldots, A_n\})\tau_1 \cup (C_2 \setminus \{\neg B_1, \ldots, \neg B_m\})\tau_2 \\ &= (C_1^* \setminus \{P(t_1, \ldots, t_k)\}) \cup (C_2^* \setminus \{\neg P(t_1, \ldots, t_k)\}) = C^*. \quad \square \end{aligned}$$

1.6.3 Věta o úplnosti

[TODO]

Úplnost

Důsledek *Nechť S' je množina všech základních instancí klauzulí formule S. Je-li $S' \vdash_R C'$ (na úrovni VL), kde C' je základní klauzule, pak existuje klauzule C a základní substituce σ t.ž. $C' = C\sigma$ a $S \vdash_R C$ (na úrovni PL).*

Důkaz Indukcí dle délky rezolučního odvození pomocí lifting lemmatu. $\square$

Věta (úplnost) *Je-li formule S nesplnitelná, je $S \vdash_R \square$.*

Důkaz Je-li S nesplnitelná, dle (důsledku) Herbrandovy věty je nesplnitelná i množina S' všech základních instancí klauzulí z S.

- Dle úplnosti rezoluční metody ve VL je $S' \vdash_R \square$ (na úrovni VL).
- Dle předchozího důsledku existuje klauzule C a substituce σ taková, že $\square = C\sigma$ a $S \vdash_R C$ (na úrovni PL).
- Jediná klauzule, jejíž instance je $\square$, je klauzule $C = \square$. $\square$

1.7 LI-rezoluce

[TODO]

Lineární rezoluce

Stejně jako ve VL, rezoluční metodu lze značně omezit (bez ztráty úplnosti).

- *Lineární důkaz* klauzule C z formule S je konečná posloupnost dvojic $(C_0, B_0), \ldots, (C_n, B_n)$ t.ž. C_0 je varianta klauzule v S a pro každé $i \leq n$
  - *i)* B_i je varianta klauzule v S nebo $B_i = C_j$ pro nějaké $j < i$, a
  - *ii)* C_{i+1} je rezolventa C_i a B_i, kde $C_{n+1} = C$.
- C je *lineárně dokazatelná* z S, psáno $S \vdash_L C$, má-li lineární důkaz z S.
- *Lineární zamítnutí* S je lineární důkaz $\square$ z S.
- S je *lineárně zamítnutelná*, pokud $S \vdash_L \square$.

Věta *S je lineárně zamítnutelná, právě když S je nesplnitelná.*

Důkaz ($\Rightarrow$) Každý lineární důkaz lze transformovat na rezoluční důkaz.
($\Leftarrow$) Plyne z úplnosti lineární rezoluce ve VL (nedokazováno), neboť lifting lemma zachovává linearitu odvození. $\square$

LI-rezoluce

Stejně jako ve VL, pro Hornovy formule můžeme lineární rezoluci dál omezit.

- *LI-rezoluce ("linear input")* z formule S je lineární rezoluce z S, ve které je každá boční klauzule B_i variantou klauzule ze (vstupní) formule S.
- Je-li klauzule C dokazatelná LI-rezolucí z S, píšeme $S \vdash_{LI} C$.
- *Hornova formule* je množina (i nekonečná) Hornových klauzulí.
- *Hornova klauzule* je klauzule obsahující nejvýše jeden pozitivní literál.
- *Fakt* je (Hornova) klauzule $\{p\}$, kde p je pozitivní literál.

- *Pravidlo* je (Hornova) klauzule s právě jedním pozitivním a aspoň jedním negativním literálem. Pravidla a fakta jsou *programové klauzule.*
- *Cíl* je neprázdná (Hornova) klauzule bez pozitivního literálu.

Věta *Je-li Hornova* T *splnitelná a* $T \cup \{G\}$ *nesplnitelná pro cíl* G*, lze* $\square$ *odvodit LI-rezolucí z* $T \cup \{G\}$ *začínající* G*.*

Důkaz Plyne z Herbrandovy věty, stejné věty ve VL a lifting lemmatu. $\square$

1.7.1 Rezoluce v Prologu

[TODO]

Program v Prologu

Program (v Prologu) je Hornova formule obsahující pouze programové klauzule, tj. fakta nebo pravidla.

files/rezolucePLprogram.pdf

Zajímá nás, zda daný existenční dotaz vyplývá z daného programu.

Důsledek *Pro program* P *a cíl* $G = \{\neg A_1, \ldots, \neg A_n\}$ *v proměnných* $X_1, \ldots, X_m$

(1) $P \models (\exists X_1) \ldots (\exists X_m)(A_1 \wedge \ldots \wedge A_n)$*, právě když*

(2) $\square$ *lze odvodit LI-rezolucí z* $P \cup \{G\}$ *začínající (variantou) cíle* G*.*

LI-rezoluce nad programem

Je-li odpověď na dotaz kladná, chceme navíc znát výstupní substituci.

Výstupní substituce σ LI-rezoluce $\square$ z $P \cup \{G\}$ začínající $G = \{\neg A_1, \ldots, \neg A_n\}$ je složení mgu v jednotlivých krocích (jen na proměnné v G). Platí,

$$P \models (A_1 \wedge \ldots \wedge A_n)\sigma.$$

files/rezolucePLprogramLI.pdf

Výstupní substituce $a)$ $X = jiri$, $b)$ $X = julie$.