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

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

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:*

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

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 $\varphi(x_1, \ldots, x_n)$ ekvivalentní svému *generálnímu uzávěru*

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

můžeme Tvrzení 1.2.3 vyslovit také takto:

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

Například v Příkladě 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! Například 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:

Lemma 1.2.12. *Mějme L-sentenci $\varphi=(\forall x_1)\dots(\forall x_n)(\exists y)\psi$ a nechť φ' je sentence $\varphi=(\forall x_1)\dots(\forall x_n)\psi(y/f(x_1,\dots,x_n))$, kde f je nový funkční symbol. Potom:*

(i) L-redukt každého modelu φ' je modelem φ, a

(ii) každý model φ lze expandovat na model φ'.

Důkaz. Nejprve dokažme část (i): Mějme model $\mathcal{A}'\models\varphi'$ a nechť $\mathcal{A}$ je jeho redukt na jazyk L. Pro každé ohodnocení proměnných e platí $\mathcal{A}\models\psi[e(y/a)]$ pro $a=(f(x_1,\dots,x_n))^{\mathcal{A}'}[e]$, tedy $\mathcal{A}\models\varphi$.

Nyní část (ii): Protože $\mathcal{A}\models\varphi$, existuje funkce $f^A:A^n\to A$ taková, že pro každé ohodnocení proměnných e platí $\mathcal{A}\models\psi[e(y/a)]$, kde $a=f^A(e(x_1),\dots,e(x_n))$. To znamená, že expanze struktury $\mathcal{A}$ vzniklá přidáním funkce f^A je modelem φ'. □

Poznámka 1.2.13. Expanze modelu ve druhé části tvrzení nemusí být (a typicky není) jednoznačná, na rozdíl od extenze o definici nového funkčního symbolu.

Aplikujeme-li předchozí lemma opakovaně (postupně pro všechny existenční kvantifikátory), získáme následující důsledek:

Důsledek 1.2.14. *Sentence φ a její skolemova varianta φ_S jsou ekvisplnitelné.*

1.2.3 Skolemova věta

V této podsekci shrneme celý postup popsaný v předchozích podsekcích. Klíčem je následující věta norského logika Thoralfa Skolema:

Věta 1.2.15 (Skolemova věta)**.** *Každá teorie má otevřenou konzervativní extenzi.*

Důkaz. Mějme L-teorii T. Každý axiom nahradíme jeho generálním uzávěrem (není-li to už sentence) a převedeme do PNF, tím získáme ekvivalentní teorii T'. Nyní nahradíme každý axiom teorie T' jeho Skolemovou variantou. Tím získáme teorii T'' v rozšířeném jazyce L'. Z Lemmatu 1.2.12 plyne, že L-redukt každého modelu T'' je modelem T', tedy T'' je extenzí T', a že každý model T' lze expandovat do jazyka L' na model T'', tedy jde o konzervativní extenzi. Teorie T'' je axiomatizovaná univerzálními sentencemi, odstraníme-li kvantifikátorové prefixy (tj. vezmeme-li jádra axiomů), získáme ekvivalentní otevřenou teorii T''', která je hledanou konzervativní extenzí. □

Ze sémantické charakterizace konzervativní extenze snadno plyne následující důsledek:

Důsledek 1.2.16. *Ke každé teorii existuje ekvisplnitelná otevřená teorie.*

Otevřenou teorii už můžeme snadno převést do CNF (vyjádřit *formulí* S v množinové reprezentaci) pomocí ekvivalentních syntaktických úprav, stejně jako ve výrokové logice (viz Sekce **??**).

1.3 Grounding

V této sekci si ukážeme, že máme-li otevřenou teorii, která je nesplnitelná, můžeme její nesplnitelnost doložit 'na konkrétních prvcích'. Co tím myslíme? Existuje konečně mnoho *základních (ground) instancí* axiomů (instancí, kde za proměnné substituujeme konstantní termy), takových, že jejich konjunkce (která neobsahuje žádnou proměnnou) je nesplnitelná.

Definice 1.3.1 (Základní instance)**.** Mějme otevřenou formuli φ ve volných proměnných $x_1, \ldots, x_n$. Řekneme, že instance $\varphi(x_1/t_1, \ldots, x_n/t_n)$ je *základní (ground) instance*, jsou-li všechny termy $t_1, \ldots, t_n$ konstantní (ground).

Příklad 1.3.2. Teorie $T = \{P(x,y) \vee R(x,y), \neg P(c,y), \neg R(x, f(x))\}$ v jazyce $L = \langle P, R, f, c\rangle$ nemá model. Můžeme to doložit následující konjunkcí základních instancí axiomů, kde za proměnnou x substituujeme konstantu c a za y konstantní term $f(c)$:

$$(P(c, f(c)) \vee R(c, f(c))) \ \wedge \ \neg P(c, f(c)) \ \wedge \ \neg R(c, f(c))$$

Tato sentence je zjevně nesplnitelná. Základní atomické sentence ($P(c, f(c))$ a $R(c, f(c))$ můžeme navíc (díky tomu, že neobsahují proměnné) chápat jako výrokové proměnné p_1, p_2, kde p_1 znamená 'platí $P(c, f(c))$' a p_2 znamená 'platí $R(c, f(c))$'. Dostáváme potom následující výrok, který lze snadno zamítnout rezolucí:

$$(p_1 \vee p_2) \wedge \neg p_1 \wedge \neg p_2$$

Tomuto procesu převedení na základní instance (a tím do výrokové logiky) říkáme 'grounding'. Za chvíli ho zformalizujeme a dokážeme *Herbrandovu větu*,[5] která říká, že taková nesplnitelná konjunkce základních instancí axiomů existuje pro každou nesplnitelnou teorii.

1.3.1 Přímá redukce do výrokové logiky

Uvědomme si nyní, že díky Herbrandově větě grounding umožňuje následující postup, byť neefektivní, jak zamítat formule rezolucí 'na úrovni výrokové logiky': Ve vstupní formuli S nahradíme každou klauzuli množinou všech jejích základních instancí (pokud žádné nejsou, tedy pokud jazyk neobsahuje konstantní symbol, jeden konstantní symbol do jazyka přidáme). Ve výsledné množině klauzulí S' chápeme atomické sentence jako výrokové proměnné, a S' zamítneme výrokovou rezolucí (o které víme, že je korektní a úplná).

Problémem tohoto přístupu je, že klauzulí v S' (základních instancí klauzulí s S může být mnoho, i nekonečně mnoho, např. kdykoliv je v jazyce alespoň jeden funkční (nekonstantní) symbol.

Příklad 1.3.3. Máme-li CNF formuli $S = \{\{P(x, y), R(x, y)\}, \{\neg P(c, y)\}, \{\neg R(x, f(x))\}\}$ v jazyce $L = \langle f, c\rangle$, nahradíme ji následující nekonečnou formulí S':

$$\begin{aligned} S' = \{&\{P(c,c), R(c,c)\}, \{P(c, f(c)), R(c, f(c))\}, \{P(f(c), c), R(f(c), c)\}, \dots, \\ &\{\neg P(c,c)\}, \{\neg P(c, f(c))\}, \{\neg P(c, f(f(c)))\}, \{\neg P(c, f(f(f(c))))\}, \dots, \\ &\{\neg R(c, f(c))\}, \{\neg R(f(c), f(f(c)))\}, \{\neg R(f(f(c)), f(f(f(c))))\}, \dots\} \end{aligned}$$

Ta je nesplnitelná, neboť obsahuje následující konečnou podmnožinu, která je nesplnitelná, což snadno ukážeme výrokovou rezolucí:

$$\{\{P(c, f(c)), R(c, f(c))\}, \{\neg P(c, f(c))\}, \{\neg R(c, f(c))\}\} \vdash_R \square$$

V Sekci 1.4 si ukážeme efektivní postup jak hledat vhodné základní instance klauzulí, pomocí tzv. *unifikace*.

1.3.2 Herbrandova věta

V této podsekci vyslovíme a dokážeme Herbrandovu větu. Budeme předpokládat, že jazyk obsahuje nějaký konstantní symbol: pokud v jazyce žádný není, jeden přidáme. Konstantní symbol potřebujeme k tomu, aby existovaly konstantní termy, a my mohli vytvořit tzv. *Herbrandův model*. Jde o konstrukci sémantického objektu (modelu) ze syntaktických objektů (konstantních termů) velmi podobnou *kanonickému modelu* (Definice **??**).[6]

[5] Francouzský matematik Jacques Herbrand pracoval na konci 20. let 20. století. Během své krátké kariéry (zemřel tragicky ve věku 23 let) objevil několik dalších důležitých výsledků, a mimo jiné formalizoval pojem rekurzivní funkce.

[6] Rozdíl je v tom, že nepřidáváme spočetně mnoho nových konstantních symbolů (vycházíme jen z konstantních symbolů, které už v jazyce jsou), a také nijak nepředepisujeme, jak mají vypadat relace modelu.

Definice 1.3.4 (Herbrandův model)**.** Mějme jazyk $L = \langle \mathcal{R}, \mathcal{F} \rangle$ s alespoň jedním konstantním symbolem. L-struktura $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}} \rangle$ je *Herbrandův model*, jestliže:

- A je množina všech konstantních L-termů (tzv. *Herbrandovo univerzum*), a
- pro každý n-ární funkční symbol $f \in \mathcal{F}$ a konstantní termy "t_1", ..., "t_n" $\in A$ platí:

$$f^{\mathcal{A}}(\text{“}t_1\text{”}, \ldots, \text{“}t_n\text{”}) = \text{“}f(t_1, \ldots, t_n)\text{”}$$

- Speciálně, pro každý konstantní symbol $c \in \mathcal{F}$ je $c^{\mathcal{A}} = \text{“}c\text{”}$.

Na interpretace relačních symbolů neklademe žádné podmínky.

Připomeňme, že uvozovky okolo termů píšeme jen neformálně, abychom jasněji odlišili termy jako syntaktické objekty (řetězce symbolů) od jejich interpretací (funkcí).

Příklad 1.3.5. Mějme jazyk $L = \langle P, f, c \rangle$, kde P je unární relační, f je binární funkční, a c konstantní symbol. Herbrandovo univerzum pro tento jazyk je množina

$$A = \{\text{“}c\text{”}, \text{“}f(c,c)\text{”}, \text{“}f(c,f(c,c))\text{”}, \text{“}f(f(c,c),c)\text{”} \ldots\}$$

Struktura $\mathcal{A} = \langle A, P^{\mathcal{A}}, f^{\mathcal{A}}, c^{\mathcal{A}} \rangle$ je Herbrandův model, jestliže $c^{\mathcal{A}} = \text{“}c\text{”}$ a funkce $f^{\mathcal{A}}$ splňuje:

- $f^{\mathcal{A}}(\text{“}c\text{”}, \text{“}c\text{”}) = \text{“}f(c,c)\text{”}$,
- $f^{\mathcal{A}}(\text{“}c\text{”}, \text{“}f(c,c)\text{”}) = \text{“}f(c,f(c,c))\text{”}$,
- $f^{\mathcal{A}}(\text{“}f(c,c)\text{”}, \text{“}c\text{”}) = \text{“}f(f(c,c),c)\text{”}$, atd.

Relace $P^{\mathcal{A}}$ může být libovolná podmnožina A.

Nyní jsme připraveni vyslovit Herbrandovu větu. Neformálně řečeno, je-li teorie splnitelná, tj. má-li model, potom má dokonce Herbrandův model, a v opačném případě najdeme nesplnitelnou konjunkci základních instancí axiomů, použitelnou pro rezoluční zamítnutí 'na úrovni výrokové logiky'.

Věta 1.3.6 (Herbrandova věta)**.** *Mějme otevřenou teorii T v jazyce L bez rovnosti a s alespoň jedním konstantním symbolem. Potom buď má T* Herbrandův model*, nebo existuje konečně mnoho základních instancí axiomů T, jejichž konjunkce je nesplnitelná.*

Důkaz. Označme jako T_{ground} množinu všech základních instancí axiomů teorie T. Zkonstruujeme systematické[7] tablo z teorie T_{ground} s položkou F$\bot$ v kořeni, ale z jazyka L, bez rozšíření o pomocné konstantní symboly na jazyk L_C.

Pokud tablo obsahuje bezespornou větev, potom je kanonický model pro tuto větev (opět bez přidání pomocných konstantních symbolů) Herbrandovým modelem T. V opačném případě máme tablo důkaz sporu, tedy teorie T_{ground}, a tím pádem i T, je nesplnitelná. Protože je tablo důkaz konečný, použili jsme v něm jen konečně mnoho základních instancí axiomů $\alpha_{\text{ground}} \in T_{\text{ground}}$. Jejich konjunkce je tedy nesplnitelná. □

Poznámka 1.3.7. Máme-li jazyk s rovností, potom nejprve teorii T rozšíříme o axiomy rovnosti na teorii T^*, a má-li T^* Herbrandův model $\mathcal{A}$, faktorizujeme ho podle kongruence $=^A$, stejně jako v případě kanonického modelu.

[7] Nebo libovolné dokončené tablo, ale tak, abychom sporné větve už neprodlužovali.

Na závěr této sekce vyslovíme dva důsledky Herbrandovy věty.

Důsledek 1.3.8. *Mějme otevřenou formuli $\varphi(x_1, \ldots, x_n)$ v jazyce L s alespoň jedním konstantním symbolem. Potom existují konstantní L-termy t_{ij} $(1 \leq i \leq m, 1 \leq j \leq n)$ takové, že sentence*

$$(\exists x_1) \ldots (\exists x_n) \varphi(x_1, \ldots, x_n)$$

je pravdivá, právě když je následující formule (výroková) tautologie:

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

Důkaz. Sentence $(\exists x_1) \ldots (\exists x_n) \varphi(x_1, \ldots, x_n)$ je pravdivá, právě když $(\forall x_1) \ldots (\forall x_n) \neg\varphi$ je nesplnitelná, neboli když $\neg\varphi$ je nesplnitelná. Tvrzení plyne z Herbrandovy věty aplikované na formuli $\neg\varphi$. □

Důsledek 1.3.9. *Mějme otevřenou teorii T v jazyce s alespoň jedním konstantním symbolem. Teorie T má model, právě když má model teorie T_{ground} sestávající ze všech základních instancí axiomů teorie T.*

Důkaz. V modelu teorie T platí všechny axiomy, tedy i všechny základní instance axiomů. Je tedy i modelem T_{ground}. Pokud T nemá model, podle Herbrandovy věty je nějaká konečná podmnožina teorie T_{ground} nesplnitelná. □

1.4 Unifikace

Místo substitucí *všech* základních termů a práce s touto novou, obrovskou a typicky nekonečnou množinou klauzulí, je lepší najít v konkrétním rezolučním kroku 'vhodnou' substituci a pracovat jen s ní. V této sekci vysvětlíme, co znamená 'vhodná' (tzv. *unifikace*) a jak ji lze hledat (pomocí *unifikačního algoritmu*).

1.4.1 Substituce

Nejprve uveďme několik příkladů 'vhodných' substitucí:

Příklad 1.4.1.

- Z klauzulí $\{P(x), Q(x, a)\}$ a $\{\neg P(y), \neg Q(b, y)\}$ získáme pomocí substituce $\{x/b, y/a\}$ klauzule $\{P(b), Q(b, a)\}$ a $\{P(b), \neg P(a)\}$, y/a, a z nich potom rezolucí klauzuli $\{P(b), \neg P(a)\}$. Mohli bychom také použít substituci $\{x/y\}$ a rezolucí přes $P(y)$ získat rezolventu $\{Q(y, a), \neg Q(b, y)\}$.

- Máme-li klauzule $\{P(x), Q(x, a), Q(b, y)\}$ a $\{\neg P(v), \neg Q(u, v)\}$, vhodnou substitucí je $\{x/b, y/a, u/b, v/a\}$; dostáváme $\{P(b), Q(b, a)\}$ a $\{\neg P(a), \neg Q(b, a)\}$, jejichž rezolventou je $\{P(b), \neg P(a)\}$.

- Podívejme se ještě na klauzule $\{P(x), Q(x, z)\}$ a $\{\neg P(y), \neg Q(f(y), y)\}$. Mohli bychom použít substituci $\{x/f(a), y/a, z/a\}$ a získat tak dvojici klauzulí $\{P(f(a)), Q(f(a), a)\}$ a $\{\neg P(a), \neg Q(f(a), a)\}$, rezolucí potom $\{P(f(a)), \neg P(a)\}$.

  Lepší ale bude využít substituce $\{x/f(z), y/z\}$, po které máme $\{P(f(z)), Q(f(z), z)\}$ a $\{\neg P(z), \neg Q(f(z), z)\}$, a rezolventu $\{P(f(z)), \neg P(z)\}$. Tato substituce je *obecnejší*, a výsledná rezolventa 'říká více' než $\{P(f(a)), \neg P(a)\}$ (ta je jejím důsledkem, ale naopak to neplatí).

Nyní zavedeme potřebnou terminologii týkající se substitucí. Substituce budeme aplikovat na termy nebo na literály (atomické formule nebo jejich negace), označme tyto dohromady jako *výrazy*.

Definice 1.4.2 (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 vyžadujeme, aby term t_i nebyl roven proměnné x_i. Substituce σ je

- *základní*, jsou-li všechny termy t_i konstantní,
- *přejmenování proměnných*, jsou-li všechny termy t_i navzájem různé proměnné.

Instance výrazu (termu nebo literálu) *Epři substituci* $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ je výraz vzniklý z E simultánním nahrazením všech výskytů proměnných x_i termy t_i, označme jej $E\sigma$. Je-li S množina výrazů, potom značíme $S\sigma = \{E\sigma \mid E \in S\}$.

Protože proměnné nahrazujeme *simultánně* pro všechny proměnné zároveň, případný výskyt proměnné x_i v termu t_j nepovede ke zřetězení substitucí.

Příklad 1.4.3. Například pro $S = \{P(x), R(y, z)\}$ a substituci $\sigma = \{x/f(y, z), y/x, z/c\}$ máme:

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

Substituce můžeme přirozeně *skládat*. Složení substitucí σ a τ, kde nejprve aplikujeme σ a potom τ, budeme zapisovat jako $\sigma\tau$. Bude tedy platit $E(\sigma\tau) = (E\sigma)\tau$, pro libovolný výraz E.

Příklad 1.4.4. Začněme opět příkladem. Máme-li výraz $E = P(x, w, u)$, a substituce

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

potom je $E\sigma = P(f(y), v, u)$ a $(E\sigma)\tau = P(f(g(x)), w, c)$. Musí tedy platit:

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

Nyní formální definice:

Definice 1.4.5 (Skládání substitucí)**.** Mějme substituce $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ a $\tau = \{y_1/s_1, \ldots, y_n/s_n\}$. *Složení substitucí* σ *a* τ je substituce

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

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

Všimněte si, že skládání substitucí není komutativní, $\sigma\tau$ je typicky zcela jiná substituce než $\tau\sigma$.

Příklad 1.4.6. Jsou-li σ a τ jako v Příkladu 1.4.4, potom:

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

Nyní ukážeme, že takto definované skládání substitucí splňuje požadovanou vlastnosti, a také že je *asociativní*. Z asociativity plyne, že nemusíme (a také nebudeme) psát závorky ve složení $\sigma\tau\varrho$, $\sigma_1\sigma_2\cdots\sigma_n$ apod.

Tvrzení 1.4.7. *Mějme substituce σ, τ, ϱ, a libovolný výraz E. Potom platí:*

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

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

Důkaz. Nechť $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ a $\tau = \{y_1/s_1, \ldots, y_m/s_m\}$. Stačí dokázat v případě, kdy výraz E je jediná proměnná, zbytek snadno plyne indukcí. (Substituce nijak nemění ostatní symboly.) Rozdělíme na tři případy:

- Je-li $E = x_i$ pro nějaké i, potom $E\sigma = t_i$ a $(E\sigma)\tau = t_i\tau = E(\sigma\tau)$, kde druhá rovnost je z definice $\sigma\tau$.
- Je-li $E = y_j$ pro nějaké j, potom $E\sigma = E$ a $(E\sigma)\tau = E\tau = s_j = E(\sigma\tau)$ opět z definice $\sigma\tau$.
- Je-li E jiná proměnná, potom $(E\sigma)\tau = E = E(\sigma\tau)$.

Tím jsme dokázali (i). Asociativitu (ii) snadno dokážeme opakovaným užitím (i). Následující platí pro každý výraz E, tedy i pro $E \in \{x_1, \ldots, x_n\}$:

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

Z toho plyne, že $(\sigma\tau)\varrho$ a $\sigma(\tau\varrho)$ jsou touž substitucí. (Podrobněji: používáme zřejmou vlastnost, že pro substituci $\pi = \{x_1/v_1, \ldots, x_n/v_n\}$ a $E = x_i$ platí $E\pi = v_i$.) □

1.4.2 Unifikační algoritmus

Které substituce jsou tedy 'vhodné'? Takové, po jejichž provedení se dané výrazy 'stanou stejnými', tj. *unifikovanými* (viz Příklad 1.4.1).

Definice 1.4.8 (Unifikace)**.** Mějme konečnou množinu výrazů $S = \{E_1, \ldots, E_n\}$. Substituce σ je *unifikace pro* S, pokud $E_1\sigma = E_2\sigma = \cdots = E_n\sigma$, neboli $S\sigma$ obsahuje jediný výraz. Pokud existuje, potom říkáme také, že S je *unifikovatelná*.

Unifikace pro S je *nejobecnější*, pokud pro každou unifikaci τ pro S existuje substituce λ taková, že $\tau = \sigma\lambda$. Všimněte si, že nejobecnějších unifikací pro S může být více, ale liší se jen přejmenováním proměnných.

Příklad 1.4.9. Uvažme množinu výrazů $S = \{P(f(x), y), P(f(a), w)\}$. Nejobecnější unifikací pro S je $\sigma = \{x/a, y/w\}$. Jinou unifikací je např. $\tau = \{x/a, y/b, w/b\}$, není ale nejobecnější, nelze z ní získat např. unifikaci $\varrho = \{x/a, y/c, w/c\}$. Unifikaci τ naopak lze získat z nejobecnější unifikace σ, a to pomocí substituce $\lambda = \{w/b\}$: $\tau = \sigma\lambda$

Nyní představíme *unifikační algoritmus*. Jeho vstupem je neprázdná, konečná množina výrazů S, a výstupem je buď nejobecnější unifikace pro S, nebo informace, že S není unifikovatelná. Algoritmus postupuje od začátku výrazů a postupně aplikuje substituce tak, aby se výrazy stávaly více podobnými. Potřebujeme následující definici:

Nechť p je první (nejlevější) pozice, na které se nějaké dva výrazy z S liší. Potom *neshoda v* S, označme $D(S)$, je množina všech podvýrazů začínajících na pozici p výrazů z S.

Příklad 1.4.10. Pro $S = \{P(x, y), P(f(x), z), P(z, f(x))\}$ je $p = 3$ a $D(S) = \{x, f(x), z\}$.

Algoritmus (Unifikační algoritmus)**.**

- **vstup**: konečná množina výrazů $S \neq \emptyset$,
- **výstup:** nejobecnější unifikace σ pro S nebo informace, že S není unifikovatelná

(0) nastav $S_0 := S$, $\sigma_0 := \emptyset$, $k := 0$

(1) pokud $|S_k| = 1$, vrať $\sigma = \sigma_0\sigma_1 \cdots \sigma_k$

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

(3) pokud ano, nastav $\sigma_{k+1} := \{x/t\}$, $S_{k+1} := S_k\sigma_{k+1}$, $k := k + 1$, a jdi na (1)

(4) pokud ne, odpověz, že S není unifikovatelná

Poznámka 1.4.11. Hledání proměnné x a termu t v kroku (2) může být relativně výpočetně náročné.

Než se pustíme do důkazu korektnosti, ukážeme si běh algoritmu na příkladě

Příklad 1.4.12. Aplikujme unifikační algoritmus na následující množinu:

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

($k = 0$) Množina $S_0 = S$ není jednoprvková, $D(S_0) = \{y, h(w), h(b)\}$ obsahuje term $h(w)$ a proměnnou y nevyskytující se v $h(w)$. Nastavíme $\sigma_1 = \{y/h(w)\}$ a $S_1 = S_0\sigma_1$, tj. máme:

$$S_1 = \{P(f(h(w), g(z)), h(b)),\ P(f(h(w), g(a)), t),\ P(f(h(b), g(z)), h(w))\}$$

($k = 1$) $D(S_1) = \{w, b\}$, $\sigma_2 = \{w/b\}$, $S_2 = S_1\sigma_2$, tj.

$$S_2 = \{P(f(h(b), g(z)), h(b)),\ P(f(h(b), g(a)), t)\}$$

($k = 2$) $D(S_2) = \{z, a\}$, $\sigma_3 = \{z/a\}$, $S_3 = S_2\sigma_3$, tj.

$$S_3 = \{P(f(h(b), g(a)), h(b)),\ P(f(h(b), g(a)), t)\}$$

($k = 3$) $D(S_3) = \{h(b), t\}$, $\sigma_4 = \{t/h(b)\}$, $S_4 = S_3\sigma_4$, tj.

$$S_4 = \{P(f(h(b), g(a)), h(b))\}$$

($k = 4$) S_4 je jednoprvková, nejobecnější unifikace pro S je následující:

$$\sigma = \sigma_1\sigma_2\sigma_3\sigma_4\{y/h(w)\}\{w/b\}\{z/a\}\{t/h(b)\} = \{y/h(b), w/b, z/a, t/h(b)\}$$

Tvrzení 1.4.13. *Unifikační algoritmus je korektní. Pro každý vstup S skončí v konečně mnoha krocích, a je-li S unifikovatelná, odpoví nejobecnější unifikaci σ, jinak odpoví, že S není unifikovatelná.*

Je-li S unifikovatelná, potom pro sestrojenou nejobecnější unifikaci σ navíc platí, že je-li τ libovolná unifikace, potom $\tau = \sigma\tau$.

Důkaz. V každém kroku k eliminujeme nějakou proměnnou, algoritmus tedy musí skončit. Pokud algoritmus skončí neúspěchem v kroku k, potom nelze unifikovat množinu S_k. Lze snadno nahlédnout, že v tom případě nelze unifikovat ani S.

Pokud algoritmus odpoví $\sigma = \sigma_0\sigma_1 \cdots \sigma_k$, zjevně jde o unifikaci. Zbývá dokázat, že je nejobecnější, k tomu stačí dokázat silnější vlastnost ('navíc') popsanou v tvrzení.

Mějme libovolnou unifikaci τ pro S. Ukážeme indukcí, že pro každé $0 \leq i \leq k$ platí:

$$\tau = \sigma_0\sigma_1 \cdots \sigma_i\tau$$

Pro $i = 0$ je $\sigma_0 = \emptyset$ a $\tau = \sigma_0\tau$ tedy platí triviálně. Předpokládejme, že to platí pro nějaké i, a dokažme pro $i + 1$. Nechť $\sigma_{i+1} = \{x/t\}$. Stačí dokázat, že pro libovolnou proměnnou u platí:

$$u\sigma_{i+1}\tau = u\tau$$

Z toho už okamžitě plyne i $\tau = \sigma_0\sigma_1 \cdots \sigma_i\sigma_{i+1}\tau$.

Je-li $u \neq x$, potom $u\sigma_{i+1} = u$, tedy i $u\sigma_{i+1}\tau = u\tau$. V případě $u = x$ označme $u\sigma_{i+1} = x\sigma_{i+1} = t$. Protože τ unifikuje množinu $S_i = S\sigma_0\sigma_1 \cdots \sigma_i$, a proměnná x i term t jsou v neshodě $D(S_i)$, musí τ unifikovat x a t. Jinými slovy, $t\tau = x\tau$, neboli $u\sigma_{i+1}\tau = u\tau$, což jsme chtěli dokázat. □

1.5 Rezoluční metoda

Chceme-li dokázat, že $T \models \varphi$, umíme díky Skolemizaci najít CNF formuli S, která je nesplnitelná, právě když je nesplnitelná teorie $T \cup \{\neg\varphi\}$, neboli právě když $T \models \varphi$. Stačí tedy najít rezoluční zamítnutí S.

V této sekci popíšeme vlastní rezoluční metodu. Většina pojmů i tvrzení bude velmi podobná výrokové logice. Jediným podstatným rozdílem bude *rezoluční pravidlo*.

1.5.1 Rezoluční pravidlo

Rezolventou dvojice klauzulí bude klauzule, kterou z nich lze odvodit aplikací *(nejobecnější) unifikace*. Nejprve příklad:

Příklad 1.5.1. Mějme klauzule $C_1 = \{P(x), Q(x, y), Q(x, f(z))\}$ a $C_2 = \{\neg P(u), \neg Q(f(u), u)\}$. Vyberme z první *oba* pozitivní literály začínající Q a ze druhé negativní literál začínající $\neg Q$. Množinu výrazů $S = \{Q(x, y), Q(x, f(z)), Q(f(u), u)\}$ lze unifikovat pomocí nejobecnější unifikace $\sigma = \{x/f(f(v)), y/f(v), z/v, u/f(v)\}$. Po aplikaci této unifikace získáme klauzule $C_1 = \{P(f(f(v))), Q(f(f(v)), f(v))\}$ a $\{\neg P(f(v)), \neg Q(f(f(v)), f(v))\}$, z nichž odvodíme klauzuli $C = \{P(f(f(v))), \neg P(f(v))$. Té budeme říkat *rezolventa* původních klauzulí C_1 a C_2.

Definice 1.5.2 (Rezoluční pravidlo)**.** Mějme klauzule C_1 a C_2 s disjunktními množinami proměnných a nechť jsou 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 $n, m \geq 1$ a množinu výrazů $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$ lze unifikovat.[8] Buď σ nejobecnější unifikace S.[9] *Rezolventa* klauzulí C_1 a C_2 je následující klauzule:

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

[8] Symbol $\sqcup$ označuje *disjunktní sjednocení*.

[9] Připomeňme, že unifikace znamená, že $A_1\sigma = A_2\sigma = \cdots = B_1\sigma = \cdots = B_m\sigma$.

Poznámka 1.5.3. Podmínku o disjunktních množinách proměnných můžeme vždy splnit, pokud přejmenujeme proměnné v jedné z klauzulí. Proč je to potřeba? Například, z klauzulí $\{\{P(x)\}, \{\neg P(f(x))\}\}$ můžeme získat prázdnou klauzuli $\square$, pokud nahradíme klauzuli $\{P(x)\}$ klauzulí $\{\{P(y)\}$. Množina výrazů $\{P(x), \neg P(f(x))\}$ ale není unifikovatelná, bez přejmenování proměnných by to tedy nešlo.

1.5.2 Rezoluční důkaz

Jakmile máme definované rezoluční pravidlo, můžeme zavést *rezoluční důkaz* a související pojmy. Definice budou stejné jako ve výrokové logice, s jedním rozdílem: dovolíme si přejmenovat proměnné v klauzulích, viz Poznámka 1.5.3.

Definice 1.5.4 (Rezoluční důkaz)**.** *Rezoluční důkaz (odvození)* klauzule C z formule S je *konečná* posloupnost klauzulí $C_0, C_1, \ldots, C_n = C$ taková, že pro každé i je

- buď $C_i = C_i'\sigma$ pro nějakou klauzuli $C_i \in S$ a přejmenování proměnných σ, nebo
- C_i je rezolventou nějakých C_j, C_k kde $j < i$ a $k < i$.

Pokud rezoluční důkaz existuje, říkáme, že C je *rezolucí dokazatelná* z S, a píšeme $S \vdash_R C$. *(Rezoluční) zamítnutí* formule S je rezoluční důkaz $\square$ z S, v tom případě je S *(rezolucí) zamítnutelná*.

Poznámka 1.5.5. Proč potřebujeme v rezolučním kroku odstranit více literálů z jedné klauzule najednou? Uvažte formuli $S = \{\{P(x), P(y)\}, \{\neg P(x), \neg P(y)\}\}$. Ta je rezolucí zamítnutelná, ale neexistuje zamítnutí, které by v každém kroku eliminovalo jen jeden literál.

Nyní si ukážeme příklad použití rezoluční metody k důkazu platnosti sentence.

Příklad 1.5.6. Nechť $T = \{\neg P(x,x), P(x,y) \to P(y,x), P(x,y) \wedge P(y,z) \to P(x,z)\}$ a nechť φ je sentence $(\exists x)\neg P(x, f(x))$. Chceme ukázat, že $T \models \varphi$. Teorie $T \cup \{\neg\varphi\}$ je ekvisplnitelná (v tomto případě dokonce ekvivalentní) s následující CNF formulí:

$$S = \{\{\neg P(x,x)\}, \{\neg P(x,y), P(y,x)\}, \{\neg P(x,y), \neg P(y,z), P(x,z)\}, \{P(x, f(x))\}\}$$

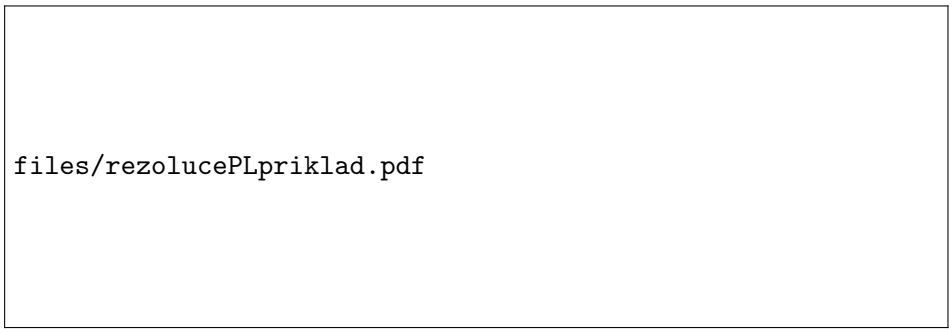
Ukážeme, že $S \vdash_R \square$. Rezolučním důkazem je například následující posloupnost:

$$\{\neg P(x,y), \neg P(y,z), P(x,z)\}, \{P(x', f(x'))\}, \{\neg P(f(x), z), P(x,z)\}, \{\neg P(x,y), P(y,x)\},$$
$$\{P(x', f(x'))\}, \{P(f(x'), x')\}, \{P(x,x)\}, \{P(x',x')\}, \square$$

Názornější je ale rezoluční strom, který je znázorněný na Obrázku 1.5.2.

1.6 Korektnost a úplnost

V této sekci dokážeme, že rezoluční metoda je i v predikátové logice korektní a úplná.



[TODO]

Obrázek 1.1: Rezoluční zamítnutí formule S z Příkladu 1.5.6. U každého rezolučního kroku je zapsána použitá unifikace.

1.6.1 Věta o korektnosti

[TODO]

Začneme důkazem korektnosti rezolučního pravidla. Princip je stejný jako u analogického pozorování ve výrokové logice. Důkaz je o trochu techničtější:

Tvrzení 1.6.1 (Korektnost rezolučního kroku)**.** *Mějme klauzule C_1, C_2 a nechť C je jejich rezolventou. Platí-li v nějaké struktuře $\mathcal{A}$ klauzule C_1 a C_2, potom v ní platí i C.*

Důkaz. Z definice rezolučního pravidla víme, že klauzule a jejich rezolventu lze vyjádřit jako $C_1 = C_1' \sqcup \{A_1, \ldots, A_n\}$, $C_2 = C_2' \sqcup \{\neg B_1, \ldots, \neg B_m\}$, a $C = C_1'\sigma \cup C_2'\sigma$, kde σ je nejobecnější unifikace množiny výrazů $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$, neboli $S\sigma = \{A_1\sigma\}$.

Protože klauzule C_1 a C_2 jsou otevřené formule platné v $\mathcal{A}$, platí v $\mathcal{A}$ i jejich instance po substituci σ tj. máme $\mathcal{A} \models C_1\sigma$ a $\mathcal{A} \models C_2\sigma$. Víme také, že $C_1\sigma = C_1'\sigma \cup \{A_1\sigma\}$ a podobně $C_2\sigma = C_2'\sigma \cup \{\neg A_1\sigma\}$.

Naším cílem je ukázat, že $\mathcal{A} \models C[e]$ pro libovolné ohodnocení proměnných e. Pokud $\mathcal{A} \models A_1\sigma[e]$, potom $\mathcal{A} \not\models \neg A_1\sigma[e]$ a musí být $\mathcal{A} \models C_2'\sigma$. Tedy i $\mathcal{A} \models C$. V opačném případě $\mathcal{A} \not\models A_1\sigma[e]$, musí tedy platit $\mathcal{A} \models C_1'\sigma$, a opět $\mathcal{A} \models C$. □

Znění i důkaz Věty o korektnosti jsou nyní stejné jako ve výrokové logice:

Věta 1.6.2 (O korektnosti rezoluce)**.** *Pokud je CNF formule S rezolucí zamítnutelná, potom je nesplnitelná.*

Důkaz. Víme, že $S \vdash_R \square$, vezměme tedy nějaký rezoluční důkaz $\square$ z S. Kdyby existoval model $\mathcal{A} \models S$, díky korektnosti rezolučního pravidla bychom mohli dokázat indukcí podle délky důkazu, že i $\mathcal{A} \models \square$, což ale není možné. □

1.6.2 Věta o úplnosti

Větu o úplnosti rezoluce v predikátové logice, totiž že nesplnitelné formule lze zamítnout rezolucí, dokážeme převedením na případ výrokové logiky. Ukážeme, že rezoluční důkaz 'na úrovni výrokové logiky' je možné 'zvednout' ('lift') na úroveň predikátové logiky.

Klíčem je následující lemma, které zaručuje takové 'zvednutí' v jednom rezolučním kroku. Jeho důkaz je poněkud technický.

Lemma 1.6.3. *Mějme klauzule C_1 a C_2 (s disjunktní množinou proměnných). Nechť $C_1^* = C_1\tau_1$ a $C_2^* = C_2\tau_2$ jsou jejich základní instance.*

Je-li C^* je rezolventou C_1^* a C_2^*, potom existuje rezolventa C klauzulí C_1 a C_2 taková, že $C^* = C\tau_1\tau_2$ je základní instancí C.

Důkaz. Nechť C^* je rezolventou C_1^* a C_2^* přes literál $P(t_1, \ldots, t_k)$. Víme, že klauzule C_1 a C_2 můžeme vyjádřit jako $C_1 = C_1' \sqcup \{A_1, \ldots, A_n\}$ 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 $\{\neg B_1, \ldots, \neg B_m\}\tau_2 = \{\neg P(t_1, \ldots, t_k)\}$.

To znamená, že $(\tau_1\tau_2)$ je unifikuje množinu výrazů $S = \{A_1, \ldots, A_n, B_1, \ldots, B_m\}$. Nyní vezměme nejobecnější unifikaci σ pro S získanou pomocí Unifikačního algoritmu. Jako C zvolme rezolventu $C = C_1'\sigma \cup C_2'\sigma$.

Zbývá ukázat, že $C^* = C\tau_1\tau_2$. Díky vlastnosti 'navíc' z Tvrzení 1.4.13 o korektnosti Unifikačního algoritmu víme, že $(\tau_1\tau_2) = \sigma(\tau_1\tau_2)$, což využíváme ve třetí rovnosti z následujícího výpočtu:

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

□

Indukcí podle délky rezolučního důkazu snadno získáme následující důsledek:

Důsledek 1.6.4. *Mějme CNF formuli S a označme jako S^* množinu všech jejích základních instancí. Pokud $S^* \vdash_R C^*$ ('na úrovni výrokové logiky') pro nějakou základní klauzuli C^*, potom existuje základní klauzule C a základní substituce σ taková, že $C^* = C\sigma$ a $S \vdash_R C$ ('na úrovni predikátové logiky').*

Nyní už je snadné dokázat úplnost:

Věta 1.6.5 (O úplnosti rezoluce)**.** *Je-li CNF formule S nesplnitelná, potom je zamítnutelná rezolucí.*

Důkaz. Označme jako S^* množinu všech základních instancí klauzulí z S. Protože je S nesplnitelná, je díky Herbrandově větě (konkrétně Důsledek 1.3.9) nesplnitelná i S^*. Z věty o úplnosti *výrokové* rezoluce víme, že $S^* \vdash_R \square$ ('na úrovni výrokové logiky'). Z lifting lemmatu (resp. z Důsledku 1.6.4) dostáváme klauzuli C a základní substituci σ takové, že $C\sigma = \square$ a $S \vdash_R C$ ('na úrovni predikátové logiky'). Ale protože prázdná klauzule $\square$ je instancí C, musí být $C = \square$. Tím jsme našli rezoluční zamítnutí $S \vdash_R \square$. □

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.*

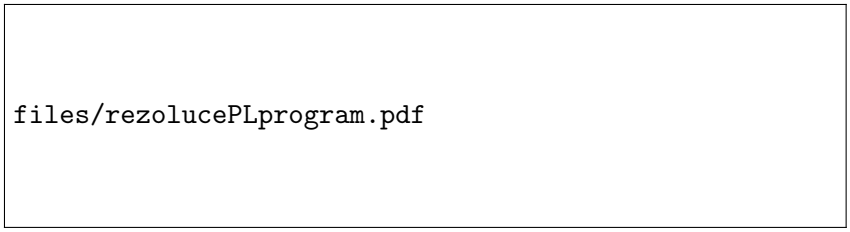
Důkaz Plyne z Herbrandovy věty, stejné věty ve VL a lifting lemmatu. $\square$

1.7.1 (draft) Rezoluce v Prologu

[TODO]

Program v Prologu

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



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*

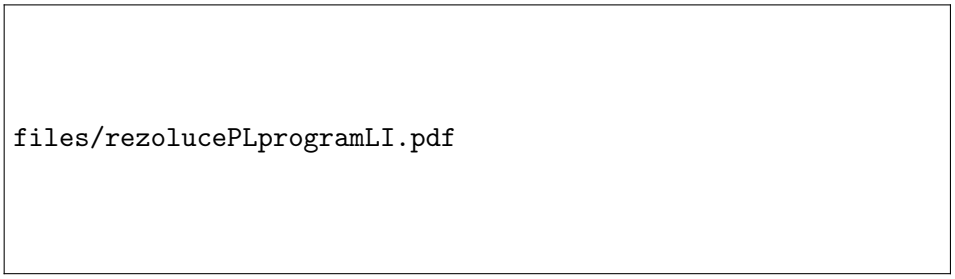
(2) $\square$ *lze odvodit LI-rezolucí z $P \cup \{G\}$ začínající (variantou) cíle G.*

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,

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



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