

# 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  $\varphi$  (tj.  $T \models \varphi$ ), začneme s teorií  $T \cup \{\neg\varphi\}$ . Tuto teorii ‘převédeme’ do CNF, a výslednou množinu klauzulí  $S$  *zamítneme* rezolucí (tj. ukážeme, že  $S \vdash_R \Box$ ) čímž ukážeme, že je nespílitelná.

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

---

<sup>1</sup>Tj.  $R(t_1, \dots, t_n)$  resp.  $t_1 = t_2$ , kde  $t_i$  jsou  $L$ -termy a  $R$  je  $n$ -ární relační symbol z  $L$ .

klauzulí.<sup>2</sup> 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  $\varphi$  *otevřené* (neobsahují-li kvantifikátory), můžeme snadno sestavit 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.<sup>3</sup>

*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 přidávanými 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).

<sup>2</sup>Jako ve výrokové logice připouštíme i nekonečné množiny klauzulí.

<sup>3</sup>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 nespílitelná, 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 nespílitelné. 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.<sup>4</sup> Výsledná CNF formule by po převedení na ‘úroveň výrokové logiky’ už byla nespílitelná.

*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 upozorníme 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  $\psi, \chi$  (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

<sup>4</sup>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  $\varphi$  je v *prenexní normální formě* (PNF), je-li tvaru

$$(Q_1x_1) \dots (Q_nx_n)\varphi'$$

kde  $Q_i$  je kvantifikátor ( $\forall$  nebo  $\exists$ ) a formule  $\varphi'$  je otevřená. Formulí  $\varphi'$  potom říkáme *otevřené jádro*  $\varphi$  a  $(Q_1x_1) \dots (Q_nx_n)$  je *kvantifikátorový prefix*.

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

Cílem této podsekce je ukázat následující tvrzení:

**Tvrzení 1.2.3** (Převod do PNF). *Ke každé formulí  $\varphi$  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 formulí  $\varphi$  nějakou podformulí  $\psi$  ekvivalentní formulí  $\psi'$ , potom je i výsledná formule  $\varphi'$  ekvivalentní formulí  $\varphi$ .*

Převod je založen na opakovaném použití následujících syntaktických pravidel:

**Lemma 1.2.5.** *Označme jako  $\overline{Q}$  kvantifikátor opačný ke  $Q$ . Nechť  $\varphi$  a  $\psi$  jsou formule, a proměnná  $x$  nechť není volná ve formulí  $\psi$ . 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).  $\square$

Všimněte si, že v pravidle  $(Qx)\varphi \rightarrow \psi \sim (\overline{Q}x)(\varphi \rightarrow \psi)$  pro vytýkání z *antecedentu* 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:

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

Všimněte si také předpokladu, že  $x$  není volná v  $\psi$ . 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  $((\forall z)P(x, z) \wedge P(y, z)) \rightarrow \neg(\exists x)P(x, y)$  do PNF. Zapišeme 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.

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

Nyní nám již nic nebrání dokázat Tvrzení 1.2.3:

*Důkaz Tvrzení 1.2.3.* Indukcí podle struktury formule  $\varphi$  s využitím Lemmatu 1.2.5 a Pozorování 1.2.4.  $\square$

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  $\varphi$  existuje ekvivalentní sentence v PNF.*

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

*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  $\varphi$  je otevřená, mohli bychom ji prostě nahradit jejím otevřeným jádrem  $\varphi$ , 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  $\varphi$  v PNF, a necht' všechny její vázané proměnné jsou různé. Necht' existenční kvantifikátory z prefixu  $\varphi$  jsou  $(\exists y_1), \dots, (\exists y_n)$  (v tomto pořadí), a necht' pro každé  $i$  jsou  $(\forall x_1), \dots, (\forall x_{n_i})$  právě všechny univerzální kvantifikátory předcházející kvantifikátor  $(\exists y_i)$  v prefixu  $\varphi$ .

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  $\varphi$  je  $L'$ -sentence  $\varphi_S$  vzniklá z  $\varphi$  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 necht'  $\varphi'$  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  $\varphi'$  je modelem  $\varphi$ , a
- (ii) každý model  $\varphi$  lze expandovat na model  $\varphi'$ .

*Důkaz.* Nejprve dokažme část (i): Mějme model  $\mathcal{A}' \models \varphi'$  a necht'  $\mathcal{A}$  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  $f^A : A^n \rightarrow A$  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  $\varphi'$ .  $\square$

*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  $\varphi$  a její skolemova varianta  $\varphi_S$  jsou ekvivalentní.*

### 1.2.3 Skolemova věta

V této podsekcí shrneme celý postup popsany 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.  $\square$

Ze sémantické charakterizace konzervativní extenze snadno plyne následující důsledek:

**Důsledek 1.2.16.** *Ke každé teorii existuje ekvivalentní 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  $\varphi$  ve volných proměnných  $x_1, \dots, x_n$ . Řekneme, že instance  $\varphi(x_1/t_1, \dots, x_n/t_n)$  je *základní (ground) instance*, jsou-li všechny termy  $t_1, \dots, t_n$  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*,<sup>5</sup> 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 zamítnat 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 ??).<sup>6</sup>

<sup>5</sup>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.

<sup>6</sup>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  $“t_1”, \dots, “t_n” \in A$  platí:

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

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

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

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

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

- $f^{\mathcal{A}}(“c”, “c”) = “f(c, c)”$ ,
- $f^{\mathcal{A}}(“c”, “f(c, c)”) = “f(c, f(c, c))”$ ,
- $f^{\mathcal{A}}(“f(c, c)”, “c”) = “f(f(c, c), c)”$ , 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 nespílitelnou konjunktci 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 konjunktce je nespílitelná.*

*Důkaz.* Označme jako  $T_{\text{ground}}$  množinu všech základních instancí axiomů teorie  $T$ . Zkonstruuje systematické<sup>7</sup> tablo z teorie  $T_{\text{ground}}$  s položkou  $F \perp$  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  $T_{\text{ground}}$ , a tím pádem i  $T$ , je nespílitelná. Protože je tablo důkaz konečný, použili jsme v něm jen konečně mnoho základních instancí axiomů  $\alpha_{\text{ground}} \in T_{\text{ground}}$ . Jejich konjunktce je tedy nespílitelná.  $\square$

*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  $=^{\mathcal{A}}$ , stejně jako v případě kanonického modelu.

<sup>7</sup>Nebo libovolné dokončené tablo, ale tak, abychom sporné větve už neprodložovali.

Na závěr této sekce vyslovíme dva důsledky Herbrandovy věty.

**Důsledek 1.3.8.** *Mějme otevřenou formuli  $\varphi(x_1, \dots, x_n)$  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*

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

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

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

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

**Důsledek 1.3.9.** *Mějme otevřenou teorii  $T$  v jazyce s alespoň jedním konstantním symbolem. Teorie  $T$  má model, právě když má model teorie  $T_{\text{ground}}$  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  $T_{\text{ground}}$ . Pokud  $T$  nemá model, podle Herbrandovy věty je nějaká konečná podmnožina teorie  $T_{\text{ground}}$  nesplnitelná.  $\square$

## 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 Substitute

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 *obecnější*, 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. Substitute 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  $\sigma = \{x_1/t_1, \dots, x_n/t_n\}$ , kde  $x_i$  jsou navzájem různé proměnné a  $t_i$  jsou termy, přičemž vyžadujeme, aby term  $t_i$  nebyl roven proměnné  $x_i$ . Substituce  $\sigma$  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*  $\sigma = \{x_1/t_1, \dots, x_n/t_n\}$  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  $\sigma$  a  $\tau$ , kde nejprve aplikujeme  $\sigma$  a potom  $\tau$ , budeme zapisovat jako  $\sigma\tau$ . Bude tedy platit  $E(\sigma\tau) = (E\sigma)\tau$ , pro libovolný výraz  $E$ .

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

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

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 substitute  $\sigma = \{x_1/t_1, \dots, x_n/t_n\}$  a  $\tau = \{y_1/s_1, \dots, y_m/s_m\}$ . *Složení substitucí*  $\sigma$  a  $\tau$  je substitute

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

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

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

*Příklad 1.4.6.* Jsou-li  $\sigma$  a  $\tau$  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 vlastnost, a také že je *asociativní*. Z asociativity plyne, že nemusíme (a také nebudeme) psát závorky ve složení  $\sigma\tau\rho$ ,  $\sigma_1\sigma_2 \cdots \sigma_n$  apod.

**Tvrzení 1.4.7.** *Mějme substituce  $\sigma$ ,  $\tau$ ,  $\varrho$ , a libovolný výraz  $E$ . Potom platí:*

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

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

*Důkaz.* Necht'  $\sigma = \{x_1/t_1, \dots, x_n/t_n\}$  a  $\tau = \{y_1/s_1, \dots, y_m/s_m\}$ . Stačí dokázat v případě, kdy výraz  $E$  je jediná proměnná, zbytek snadno plyne indukcí. (Substituce nijak nemění ostatní symboly.)

Je-li  $E = x_i$  pro nějaké  $i$ , potom  $E\sigma = t_i$  a  $(E\sigma)\tau = t_i\tau = E(\sigma\tau)$ .

[TODO]

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

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

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

## 1.4.2 Unifikační algoritmus

[TODO]

### Unifikace

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

- *Unifikace* pro  $S$  je substituce  $\sigma$  taková, že  $E_1\sigma = E_2\sigma = \dots = E_n\sigma$ , tj.  $S\sigma$  je singleton.
- $S$  je *unifikovatelná*, pokud má unifikaci.
- Unifikace  $\sigma$  pro  $S$  je *nejobecnější unifikace (mgu)*, pokud pro každou unifikaci  $\tau$  pro  $S$  existuje substituce  $\lambda$  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  $\lambda = \{w/b\}$ .  $\tau$  není mgu, nelze z ní získat unifikaci  $\varrho = \{x/a, y/c, w/c\}$ .

*Pozorování* Jsou-li  $\sigma$ ,  $\tau$  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  $\sigma$  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  $\sigma$  pro  $S$  nebo pozná, že  $S$  není unifikovatelná. (\*) Navíc, pro každou unifikaci  $\tau$  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  $\sigma$  evidentně unifikace pro  $S$ .
- Dokážeme-li, že  $\sigma$  má vlastnost (\*), je  $\sigma$  nejobecnější unifikace pro  $S$ .

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

## 1.5 Rezoluční metoda

[TODO]

### 1.5.1 Rezoluční pravidlo

[TODO]

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

$$C_1 = C'_1 \sqcup \{A_1, \dots, A_n\}, \quad C_2 = C'_2 \sqcup \{\neg B_1, \dots, \neg B_m\},$$

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

$$C = C'_1\sigma \cup C'_2\sigma,$$

kde  $\sigma$  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  $C_0, \dots, C_n = C$  taková, že pro každé  $i \leq n$  je  $C_i = C'_i \sigma$ , kde  $C'_i \in S$  a  $\sigma$  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  $T = \{\neg P(x, x), P(x, y) \rightarrow P(y, x), P(x, y) \wedge P(y, z) \rightarrow P(x, z)\}$ .

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

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

files/rezolutePLpriklad.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  $\mathcal{A}$ ,

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

*Důkaz* Nechť  $C_1 = C'_1 \sqcup \{A_1, \dots, A_n\}$ ,  $C_2 = C'_2 \sqcup \{\neg B_1, \dots, \neg B_m\}$ ,  $\sigma$  je nejobecnější unifikace pro  $S = \{A_1, \dots, A_n, B_1, \dots, B_m\}$  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  $\mathcal{A} \models C'_1\sigma[e]$  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  $P(t_1, \dots, t_k)$ .

- Pak lze psát  $C_1 = C'_1 \sqcup \{A_1, \dots, A_n\}$  a  $C_2 = C'_2 \sqcup \{\neg B_1, \dots, \neg B_m\}$ , kde  $\{A_1, \dots, A_n\}\tau_1 = \{P(t_1, \dots, t_k)\}$  a  $\{\neg B_1, \dots, \neg B_m\}\tau_2 = \{\neg P(t_1, \dots, t_k)\}$ .
- Tedy  $(\tau_1\tau_2)$  unifikuje  $S = \{A_1, \dots, A_n, B_1, \dots, B_m\}$  a je-li  $\sigma$  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  $\sigma$  a tedy

$$\begin{aligned} C\tau_1\tau_2 &= (C'_1\sigma \cup C'_2\sigma)\tau_1\tau_2 = C'_1\sigma\tau_1\tau_2 \cup C'_2\sigma\tau_1\tau_2 = C'_1\tau_1 \cup C'_2\tau_2 \\ &= (C_1 \setminus \{A_1, \dots, A_n\})\tau_1 \cup (C_2 \setminus \{\neg B_1, \dots, \neg B_m\})\tau_2 \\ &= (C_1^* \setminus \{P(t_1, \dots, t_k)\}) \cup (C_2^* \setminus \{\neg P(t_1, \dots, t_k)\}) = C^*. \quad \square \end{aligned}$$



### 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  $\sigma$  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  $S$  nespílitelná, je  $S \vdash_R \square$ .*

*Důkaz* Je-li  $S$  nespílitelná, dle (důsledku) Herbrandovy věty je nespílitelná 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  $\sigma$  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  $(C_0, B_0), \dots, (C_n, B_n)$  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 *nespílitelná*.

*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.

`files/rezolucePLprogram.pdf`

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

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

(1)  $P \models (\exists X_1) \dots (\exists X_m)(A_1 \wedge \dots \wedge A_n)$ , 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  $\sigma$  LI-rezoluce  $\square$  z  $P \cup \{G\}$  začínající  $G = \{\neg A_1, \dots, \neg A_n\}$  je složení mgu v jednotlivých krocích (jen na proměnné v  $G$ ). Platí,

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

files/rezolucePLprogramLI.pdf

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