NAIL062 Výroková a predikátová logika: zápisky z přednášky

Jakub Bulín[1]

Zimní semestr 2024

[1]KTIML MFF UK, jakub.bulin@mff.cuni.cz

Přednáška je postavena na učebnici *A. Nerode, R. Shore: Logic for Applications* [3], některé části jsou z knihy *M. Ben-ari: Mathematical Logic for Computer Science* [1]. Struktura kurzu i převážná většina obsahu jsou převzaty z přednášek Petra Gregora z předchozích let [2], viz také anglická skripta Martina Piláta [4]. Najdete-li překlepy nebo jiné chyby, případně těžko srozumitelné části, prosím, napište mi.

Obsah

Kapitola 1

Úvod do logiky

Slovo *logika* se používá ve dvou významech:

- soubor principů, které jsou základem uspořádání prvků nějakého systému (např. počítačového programu, elektronického zařízení, komunikačního protokolu)
- uvažování prováděné podle striktních pravidel zachovávajících platnost

V informatice se tyto dva významy setkávají: nejprve formálně popíšeme daný systém, a poté o něm *formálně uvažujeme* (v praktických aplikacích automaticky), tj. odvozujeme platné *inference* o systému, za použití nějakého *(formálního) dokazovacího systému*.

Mezi praktické aplikace logiky v informatice patří například software verification, logic programming, SAT solving, automated reasoning, database theory, knowledge-based representation, a mnoho dalších. Kromě toho je logika (ve větší míře než matematika) základním nástrojem pro popis teoretické informatiky.

1.1 Výroková logika

Nyní si ukážeme logiku v akci na dvou příkladech ze života (hledače pokladů, a teoretického informatika):

1.1.1 Příklad: Hledání pokladu

Příklad 1.1.1. Při hledání pokladu v dračí sluji jsme narazili na rozcestí dvou chodeb. Víme, že na konci každé chodby je buď poklad, nebo drak, ale ne obojí. Trpaslík, kterého jsme na rozcestí potkali, nám řekl, že: "Alespoň jedna z těch dvou chodeb vede k pokladu", a po dalším naléhání (a menším úplatku), ještě řekl, že "První chodba vede k drakovi." Je známo, že trpaslíci, které člověk potká v dračí sluji, buď vždy mluví pravdu, nebo vždy lžou. Kterou cestou se máme vydat?

1.1.2 Formalizace ve výrokové logice

Začneme tím, že situaci a naše znalosti formalizujeme ve výrokové logice. *Výrok* je tvrzení, kterému můžeme přiřadit pravdivostní hodnotu: *pravdivý* (*True*, 1), nebo *lživý* (*False*, 0). Některé výroky lze vyjádřit pomocí jednodušších výroků a logických spojek, např. "(Trpaslík

lže,) *právě když* (druhá chodba vede k drakovi.)" nebo "(První chodba vede k pokladu) *nebo* (první chodba vede k drakovi.)" Pokud výrok takto rozložit nelze, říká se mu *prvovýrok*, *atomický výrok*, nebo *výroková proměnná*.

Popíšeme tedy celou situaci pomocí *výrokových proměnných*. Můžeme si je také představit jako jednoduché zjišťovací (ano/ne) otázky, na které musíme znát odpověď, abychom věděli vše o dané situaci. Jako naše výrokové proměnné zvolme "Poklad je v první chodbě." (označme p_1), a "Poklad je v druhé chodbě." (p_2). V úvahu přichází i jiné výrokové proměnné, např. "V první chodbě je drak." (d_1) nebo "Trpaslík mluví pravdu." (t). Ty lze ale vyjádřit pomocí $\{p_1, p_2\}$, např. platí t, právě když neplatí p_1. Tj. známe-li pravdivostní hodnoty p_1, p_2, pravdivostní hodnoty d_1, t jsou jednoznačně určené. A menší počet výrokových proměnných znamená menší prohledávací prostor.

Vyjádříme nyní všechny naše znalosti jako (*složené*) výroky a zapíšeme je ve formálním zápisu v *jazyce* výrokové logiky nad množinou prvovýroků $\mathbb{P} = \{p_1, p_2\}$, za použití symbolů reprezentujících logické spojky $\neg$ ("neplatí X", *negace*), $\wedge$ ("X a Y", *konjunkce*), $\vee$ ("X nebo Y", *disjunkce*), $\rightarrow$ ("pokud X, potom Y", *implikace*), $\leftrightarrow$ ("X, právě když Y", *ekvivalence*) a závorek (,). Zde je dobré zmínit, že disjunkce není exkluzivní, tj. "X nebo Y" platí i pokud platí jak X tak Y, a implikace je čistě logická: "pokud X, potom Y" platí kdykoliv neplatí X nebo platí Y.

Informace o tom, že v chodbě je poklad nebo drak, ale ne obojí, už je zakódovaná v naší volbě výrokových proměnných: přítomnost draka je totéž co absence pokladu. Tvrzení trpaslíka, že "První chodba vede k drakovi." tedy vyjádříme jako "Neplatí, že poklad je v první chodbě.", formálně $\neg p_1$. Tvrzení, že "Alespoň jedna z těch dvou chodeb vede k pokladu." vyjádříme jako "Poklad je v první chodbě nebo poklad je v druhé chodbě.", formálně $p_1 \vee p_2$. Informaci, že trpaslíci buď vždy mluví pravdu, nebo vždy lžou, si přeložíme tak, že buď platí oba naše výroky, nebo platí negace obou našich výroků, formálně:

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

Označme tento výrok jako φ (od slova "formule", výrokům se někdy také říká *výrokové formule*). V našem příkladě lze všechny informace vyjádřit jediným výrokem, v praxi ale často potřebujeme výroků více, někdy i nekonečně mnoho (například pokud chceme popsat výpočet nějakého programu a nevíme apriori kolik kroků bude mít), potom popíšeme situaci pomocí množiny výroků, tzv. *teorie*, zde $T = \{\varphi\}$. Výrokům z T říkáme také *axiomy* teorie T[1].

1.1.3 Modely a důsledky

Jsou naše informace dostačující k určení, zda je v některé konkrétní chodbě poklad? Jinými slovy, ptáme se, zda je jeden z výroků p_1, p_2 logickým *důsledkem* výroku φ, resp. teorie T. Co to znamená?

Představme si, že existuje více různých "světů" lišících se v tom, co je na konci první a na konci druhé chodby. Například, v jednom ze světů je na konci první chodby poklad a na konci druhé chodby drak. Tento svět můžeme popsat pomocí pravdivostního ohodnocení výrokových proměnných: $p_1 = 1, p_2 = 0$. Takovému ohodnocení říkáme *model* jazyka $\mathbb{P} = \{p_1, p_2\}$ a zapisujeme ho zkráceně jako $v = (1, 0)$ (v od slova "valuation"). Celkem tedy máme čtyři různé světy, popsané *modely jazyka*:

$$\mathrm{M}_{\mathbb{P}} = \{(0,0), (0,1), (1,0), (1,1)\}.$$

[1] Terminologie v logice často pochází z jejích aplikace v matematice.

Je svět popsaný modelem $v = (1, 0)$ konzistentní s informacemi, které máme, tj. *platí* v něm výrok φ, resp. teorie T? Pravdivostní hodnotu (složeného) výroku φ v modelu v, označme ji $v(\varphi)$, můžeme snadno zjistit: Víme, že $v(p_1) = 1$ a $v(p_2) = 0$, takže $v(\neg p_1) = 0$, a také $v(\neg p_1 \wedge (p_1 \vee p_2)) = 0$ (jde o konjunkci dvou výroků, a první z konjunktů je v modelu v nepravdivý). Podobně $v(p_1 \vee p_2) = 1$ (neboť $v(p_1) = 1$), takže $v(\neg(p_1 \vee p_2)) = 0$, a $v(\neg(\neg p_1) \wedge \neg(p_1 \vee p_2)) = 0$. Výrok φ je disjunkcí dvou výroků, z nichž ani jeden v modelu v neplatí, tedy $v(\varphi) = 0$.

Bystrý čtenář jistě vidí stromovou strukturu výroku φ a postupné vyhodnocování $v(\varphi)$ od listů směrem ke kořeni. Formální definice představíme v příští kapitole.

Podobně určíme pravdivostní hodnoty výroku φ v ostatních modelech. Zjistíme, že množina *modelů výroku* φ (resp. *modelů teorie* T), tj. množina všech modelů jazyka, ve kterých platí φ (resp. všechny výroky z teorie T), je

$$\mathrm{M}_{\mathbb{P}}(\varphi) = \mathrm{M}_{\mathbb{P}}(T) = \{(0, 1)\}.$$

Vidíme, že naše informace jednoznačně určují model $(0, 1)$, tedy svět, ve kterém je v první chodbě drak a ve druhé poklad. Obecně modelů může být více, stačí nám vědět, že v každém modelu φ, resp. T, platí výrok p_2, tedy že p_2 je *důsledkem* teorie T; také říkáme, že p_2 *platí* v teorii T.

1.1.4 Dokazovací systémy

Postup, který jsme zvolili, je velmi neefektivní. Máme-li n výrokových proměnných[2], existuje 2^n modelů jazyka a není prakticky možné ověřovat platnost teorie v každém z nich. Na řadu přichází tzv. *dokazovací systémy*. V daném dokazovacím systému je *důkaz* výroku ψ z teorie T přesně, formálně definovaný syntaktický objekt, který v sobě zahrnuje snadno (mechanicky) ověřitelný "důkaz" (důvod), proč ψ platí v T, a který můžeme hledat (pomocí počítače) čistě na základě struktury výroku ψ a axiomů teorie T ("syntaxe"), tj. aniž bychom se museli zabývat modely ("sémantikou").

Po dokazovacím systému chceme dvě vlastnosti:

- *korektnost*, tj. pokud máme důkaz ψ z T, potom ψ platí v T, a
- *úplnost*, tj. pokud ψ platí v T, potom existuje důkaz ψ z T,

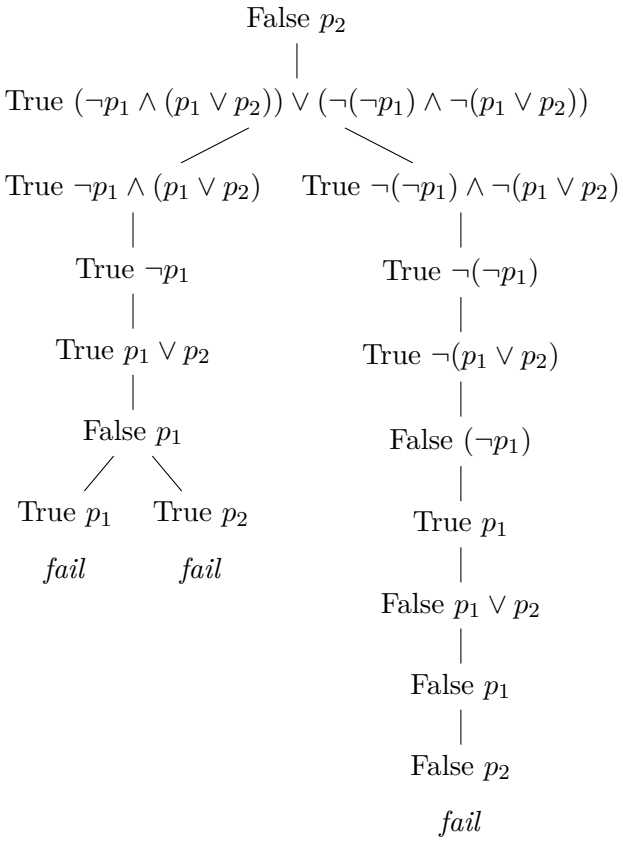
přičemž korektnost je nutností (bez ní nemá smysl důkazy hledat), a úplnost je dobrá vlastnost, ale efektivní dokazovací systém může být užitečný, i pokud v něm nelze dokázat vše, co platí.

Zde stručně nastíníme dva dokazovací systémy: *metodu analytického tabla* a *rezoluční metodu*. Později budou představeny formálně, a u obou si dokážeme i jejich korektnost a úplnost. Oba tyto dokazovací systémy jsou založeny na *důkazu sporem*, tj. předpokládají platnost axiomů z T a negace výroku ψ, a snaží se dojít ke sporu.

1.1.5 Tablo metoda

V metodě analytického tabla je důkazem *tablo*: strom jehož vrcholy jsou označkované předpoklady o platnosti výroků. Podívejme se na příklad tabla na obrázku 1.1.

[2] V praxi máme běžně tisíce proměnných.



Obrázek 1.1: Tablo důkaz výroku p_2 z teorie T

Začneme předpokladem, že neplatí výrok p_2 (neboť dokazujeme sporem). Poté připojíme platnost všech axiomů teorie T (v našem případě je jen jeden: výrok φ sestrojený výše). Dále budujeme tablo tak, že zjednodušujeme výroky v předpokladech, a to podle jistých pravidel, která zaručují následující invariant:

> *Každý model teorie T, ve kterém neplatí p_2, se musí* shodovat *s některou z větví tabla (tj. splňovat všechny předpoklady na dané větvi).*

Výrok φ je disjunkcí dvou výroků, $\varphi = \varphi_1 \vee \varphi_2$. Pokud tedy platí v nějakém modelu, potom buď v tomto modelu platí φ_1, nebo v něm platí φ_2. Rozvětvíme strom podle těchto dvou možností. V dalším kroku máme předpoklad o pravdivosti výroku $\neg p_1 \wedge (p_1 \vee p_2)$. V tom případě musí platit jak $\neg p_1$, tak $p_1 \vee p_2$, připojíme tedy na konec větve oba tyto předpoklady. Pravdivost $\neg p_1$ znamená lživost p_1, a tak dále.

Takto postupujeme, dokud je možné výroky v předpokladech zjednodušit, tj. dokud to nejsou jen výrokové proměnné. Pokud na jedné větvi najdeme dvojici opačných předpokladů o nějakém výroku ψ, tj. že platí, a zároveň že neplatí, víme, že se s touto větví nemůže shodovat žádný model. Takové větvi říkáme *sporná*. Protože dokazujeme sporem, důkaz je takové tablo, ve kterém je každá větev sporná. Tím je zaručeno, že neexistuje model T, ve kterém neplatí p_2. Z toho plyne, že p_2 platí v každém modelu T, neboli je to důsledek T, což jsme chtěli dokázat.

Zatím se spokojíme s pochopením základní myšlenky této metody, detaily představíme později, v Kapitole 4.

1.1.6 Rezoluční metoda

Není těžké naprogramovat systematické hledání tablo důkazu. V praxi se ale používá jiný dokazovací systém, který má mnohem jednodušší a efektivnější implementaci: tzv. *rezoluční metoda*. Tato metoda pochází z roku 1965, a je základem většiny *systémů automatického dokazování*, *SAT solverů*, nebo třeba interpreterů jazyka Prolog (viz Podsekce 8.7.2).

Rezoluční metoda je založena na faktu, že každý výrok lze ekvivalentně vyjádřit ve speciálním tvaru, v tzv. *konjunktivní normální formě (CNF)*. *Literál* je výroková proměnná p nebo její negace $\neg p$ (tj. literály jsou výroky, které jen určují hodnotu jedné výrokové proměnné). Disjunkci několika literálů, např. $p \vee \neg q \vee \neg r$, říkáme *klauzule*. A výrok je v CNF, pokud je konjunkcí klauzulí. Ke každému výroku ψ existuje *ekvivalentní* výrok ψ' v CNF. Ekvivalentní znamená mající stejný význam (stejné modely), píšeme $\psi \sim \psi'$. Později si ukážeme dvě metody převodu do CNF, nyní jen na našem příkladě: ve výroku

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

nejprve nahradíme $\neg(\neg p_1) \sim p_1$ a $\neg(p_1 \vee p_2) \sim (\neg p_1 \wedge \neg p_2)$:

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

a dále opakovaně použijeme *distributivitu* $\vee$ vůči $\wedge$ (představte si, že $\vee$ je operace násobení a $\wedge$ je operace sčítání):

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

Tento výrok už je v CNF, ale dále ho zjednodušíme: vynecháme z klauzulí duplicitní literály, a uvědomíme si, že obsahuje-li klauzule dvojici opačných literálů $p, \neg p$, je to *tautologie* (platí v každém modelu) a proto ji můžeme odstranit. Dostáváme CNF výrok

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

který je ekvivalentní původnímu výroku ϕ Protože chceme dokázat p_2 sporem, přidáme ještě klauzuli $\neg p_2$:

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

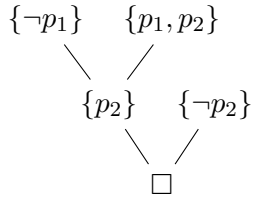
Výrok p_2 platí v teorii T, právě když je tento CNF výrok *nesplnitelný* (nemá žádný model). Výroky v CNF budeme zapisovat také v *množinové reprezentaci*[3]:

$$\{\{\neg p_1\}, \{\neg p_1, \neg p_2\}, \{p_1, p_2\}, \{\neg p_2\}\}$$

Rezoluční pravidlo říká, že pokud máme dvojici klauzulí, z nichž jedna obsahuje literál p a druhá opačný literál $\neg p$, potom z nich logicky plyne také jejich *rezolventa*: klauzule vzniklá odstraněním literálu p z první a $\neg p$ z druhé klauzule a sjednocením zbylých literálů. Například, z $p \vee \neg q \vee \neg r$ a $\neg p \vee \neg q \vee s$ můžeme odvodit rezolventu $\neg q \vee \neg r \vee s$. *Rezoluční zamítnutí* formule v CNF je potom posloupnost klauzulí, která končí *prázdnou klauzulí* $\square$ (znamenající spor) a kde každá klauzule je buď z dané formule, nebo je rezolventou nějakých dvou předchozích klauzulí. V našem případě:

$$\{\neg p_1\}, \{p_1, p_2\}, \{p_2\}, \{\neg p_2\}, \square$$

Třetí klauzule je rezolventou první a druhé, pátá je rezolventou třetí a čtvrté. Rezoluci lze také přirozeně znázornit pomocí *rezolučního stromu*, kde listy jsou klauzule z dané formule, a vnitřní vrcholy jsou rezolventy svých potomků:



Pokud by formule měla model, musely by v něm platit její klauzule, tedy i postupně všechny rezolventy v posloupnosti, a nakonec prázdná klauzule. Ta ale neplatí v žádném modelu. Máme tedy důkaz sporem a víme, že ve druhé chodbě najdeme poklad.

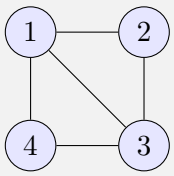
Detaily rezoluční metody představíme v Kapitole 5.

1.1.7 Příklad: Barvení grafů

Ve druhém příkladu se trochu přiblížíme aplikacím. Na rozdíl od logických hádanek ve formě slovních úloh, různé varianty problému barvení grafů se objevují v rozmanitých úlohách z praxe, od rozvrhovacích problémů přes návrh fyzických a síťových systémů po zpracování obrazu.

[3] V praktické implementaci bychom mohli použít seznam klauzulí, kde každá klauzule je seznam (unikátních) literálů v nějakém zvoleném uspořádání. Představte si opět tisíce výrokových proměnných a klauzulí.

Příklad 1.1.2. Najděte vrcholové obarvení následujícího grafu třemi barvami, tj. přiřaďte vrcholům barvy R, G, B tak, aby žádná hrana nebyla monochromatická.



Graf si reprezentujeme jako množinu vrcholů a množinu hran, kde každá hrana je dvojice vrcholů. Lépe se nám bude pracovat s uspořádanými dvojicemi, zvolíme tedy (libovolnou) orientaci hran.

$$\mathcal{G} = \langle V; E\rangle = \langle \{1,2,3,4\}; \{(1,2),(1,3),(1,4),(2,3),(3,4)\}\rangle$$

Začneme opět formalizací ve výrokové logice. Označme si množinu barev jako $C = \{R, G, B\}$. Přirozená volba výrokových proměnných je "vrchol v má barvu c", označme p_v^c, pro každý vrchol $v \in V$ a každou barvu $c \in C$. Naše (uspořádaná) množina výrokových proměnných má 12 prvků:

$$\mathbb{P} = \{p_v^c \mid c \in C, v \in V\} = \{p_1^R, p_1^G, p_1^B, p_2^R, p_2^G, p_2^B, p_3^R, p_3^G, p_3^B, p_4^R, p_4^G, p_4^B\}$$

Máme celkem $|\,\mathrm{M}_{\mathbb{P}}\,| = 2^{12} = 4096$ modelů jazyka (reprezentovaných 12-dimenzionálními 0–1 vektory). Většinu z nich nelze interpretovat jako obarvení grafu. Například $v = (1,1,0,0,\ldots,0)$ říká, že vrchol 1 je obarvený červeně, a také zeleně. Začneme tedy teorií vyjadřující, že každý vrchol má nejvýše jednu barvu. Existuje více způsobů, jak to můžeme vyjádřit. My řekneme pro každý vrchol, že nesmí mít (alespoň) jednu z každé dvojice barev. Tím dostaneme teorii v CNF:

$$\begin{aligned} T_1 = \{&(\neg p_1^R \vee \neg p_1^G) \wedge (\neg p_1^R \vee \neg p_1^B) \wedge (\neg p_1^G \vee \neg p_1^B),\\ &(\neg p_2^R \vee \neg p_2^G) \wedge (\neg p_2^R \vee \neg p_2^B) \wedge (\neg p_2^G \vee \neg p_2^B),\\ &(\neg p_3^R \vee \neg p_3^G) \wedge (\neg p_3^R \vee \neg p_3^B) \wedge (\neg p_3^G \vee \neg p_3^B),\\ &(\neg p_4^R \vee \neg p_4^G) \wedge (\neg p_4^R \vee \neg p_4^B) \wedge (\neg p_4^G \vee \neg p_4^B)\}\\ = \{&(\neg p_v^R \vee \neg p_v^G) \wedge (\neg p_v^R \vee \neg p_v^B) \wedge (\neg p_v^G \vee \neg p_v^B) \mid v \in V\} \end{aligned}$$

Teorii T_1 bychom mohli říkat teorie *částečných* vrcholových obarvení grafu $\mathcal{G}$. Teorie T_1 má $|\,\mathrm{M}_{\mathbb{P}}(T_1)| = 4^4 = 2^8 = 256$ modelů. (Proč?) Pokud chceme úplná obarvení, přidáme podmínku, že každý vrchol má alespoň jednu barvu.[4]

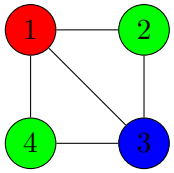
$$\begin{aligned} T_2 &= T_1 \cup \{p_1^R \vee p_1^G \vee p_1^B, p_2^R \vee p_2^G \vee p_2^B, p_3^R \vee p_3^G \vee p_3^B, p_4^R \vee p_4^G \vee p_4^B\}\\ &= T_1 \cup \{p_v^R \vee p_v^G \vee p_v^B \mid v \in V\}\\ &= T_1 \cup \{\bigvee_{c\in C} p_v^c \mid v \in V\} \end{aligned}$$

[4]Symbol $\bigvee$ používáme podobně jako symboly $\sum$ pro součet a $\prod$ pro součin: ke zjednodušení zápisu výroku, který je ve formě disjunkce. Např. pokud $v = 1$, potom $\bigvee_{c\in C} p_v^c$ reprezentuje výrok $p_1^R \vee p_1^G \vee p_1^B$. Analogicky $\bigwedge$ pro konjunkci.

Teorie T_2 má $3^4 = 81$ modelů. Jde o *extenzi* teorie T_1, neboť každý důsledek teorie T_1 platí i v teorii T_2. Platí dokonce, že $\mathrm{M}_{\mathbb{P}}(T_2) \subseteq \mathrm{M}_{\mathbb{P}}(T_1)$.[5] Zbývá přidat podmínku zakazující monochromatické hrany. Pro každou hranu a každou barvu specifikujeme, že alespoň jeden z vrcholů hrany nesmí mít danou barvu. Zde pro názornost naposledy napíšeme úplný seznam výroků, nadále budeme využívat zkráceného zápisu.

$$\begin{aligned} T_3 = T_2 \cup \{&(\neg p_1^R \vee \neg p_2^R) \wedge (\neg p_1^G \vee \neg p_2^G) \wedge (\neg p_1^B \vee \neg p_2^B),\\ &(\neg p_1^R \vee \neg p_3^R) \wedge (\neg p_1^G \vee \neg p_3^G) \wedge (\neg p_1^B \vee \neg p_3^B),\\ &(\neg p_1^R \vee \neg p_4^R) \wedge (\neg p_1^G \vee \neg p_4^G) \wedge (\neg p_1^B \vee \neg p_4^B),\\ &(\neg p_2^R \vee \neg p_3^R) \wedge (\neg p_2^G \vee \neg p_3^G) \wedge (\neg p_2^B \vee \neg p_3^B),\\ &(\neg p_3^R \vee \neg p_4^R) \wedge (\neg p_3^G \vee \neg p_4^G) \wedge (\neg p_3^B \vee \neg p_4^B)\}\\ = T_2 \cup \{ &\bigwedge_{c \in C} (\neg p_u^c \vee \neg p_v^c) \mid (u,v) \in E\} \end{aligned}$$

Výsledná teorie T_3 je *splnitelná* (má model), právě když graf $\mathcal{G}$ je 3-obarvitelný. Má 6 modelů jednoznačně odpovídajících 3-obarvení grafu $\mathcal{G}$. Model $v = (1,0,0,0,1,0,0,0,1,0,1,0)$ odpovídá následujícímu obarvení, ostatní obarvení získáme permutací barev.



Jakmile máme teorii T_3 formalizující 3-obarvení grafu $\mathcal{G}$, můžeme snadno řešit související otázky, například najít všechna obarvení, ve kterých je vrchol 1 modrý a vrchol 2 zelený: odpovídají modelům teorie $T_3 \cup \{p_1^B, p_2^G\}$. Nebo můžeme dokázat, že vrcholy 2 a 4 musejí být obarveny stejnou barvou. Můžeme použít tablo metodu: v kořeni tabla bude předpoklad

$$\text{False } (p_2^R \wedge p_4^R) \vee (p_2^G \wedge p_4^G) \vee (p_2^B \wedge p_4^B)$$

Nebo můžeme najít rezoluční zamítnutí teorie vzniklé převedením axiomů teorie T_3 do CNF, a přidáním CNF ekvivalentu *negace* výroku $(p_2^R \wedge p_4^R) \vee (p_2^G \wedge p_4^G) \vee (p_2^B \wedge p_4^B)$ (neboť jde o důkaz sporem, tedy pro spor předpokládáme, že *nemají* stejnou barvu).

1.2 Predikátová logika

Nyní si velmi stručně a neformálně představíme *predikátovou logiku*. Predikátová logika se zabývá vlastnostmi objektů a vztahy mezi objekty. Například:

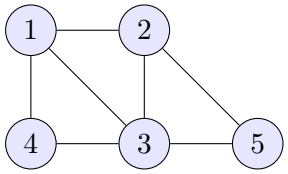
Všichni lidé jsou smrtelní.
Sókratés je člověk.
Sókratés je smrtelný.

Ve skutečnosti výroková logika vznikla později (asi o století) než Aristotelova predikátová logika, a byla poté na dlouho převážně zapomenuta.

[5] Zde vidíme typickou ukázku antimonotónního vztahu tzv. *Galoisovy korespondence*: čím více vlastností (výroků) požadujeme, tím méně objektů (modelů) splňuje tyto vlastnosti.

1.2.1 Nevýhody formalizace ve výrokové logice

Nevýhodou formalizace našeho problému o barvení grafů ve výrokové logice je fakt, že výsledná teorie T_3 je poměrně velká, a navíc byla vytvořena ad hoc pro graf $\mathcal{G}$. Představme si, že potřebujeme graf $\mathcal{G}$ změnit, například přidáním vrcholu 5 spojeného hranami s vrcholy 2 a 3:



Abychom byli schopni formalizovat nový problém, musíme přidat do našeho jazyka tři nové výrokové proměnné: $\mathbb{P}' = \mathbb{P} \cup \{p_5^R, p_5^G, p_5^B\}$, a vytvořit nové teorie T_1', T_2', T_3' přidáním axiomů týkajících se vrcholu 5 a hran $(2,5), (3,5)$. Problémem je, že strukturu grafu $\mathcal{G}$ a přirozené vlastnosti jako "z vrcholu u vede hrana do vrcholu v", nebo "vrchol u je zelený" jsme ('natvrdo', 'nepřirozeně') zakódovali do 0–1 proměnných. Tento nedostatek odstraňuje *predikátová logika.*

1.2.2 Stručné představení predikátové logiky

Modelem v predikátové logice není 0–1 vektor, ale *struktura.* Příkladem struktur jsou naše (orientované) grafy:

$$\mathcal{G} = \langle V^{\mathcal{G}}; E^{\mathcal{G}} \rangle = \langle \{1,2,3,4\}; \{(1,2),(1,3),(1,4),(2,3),(3,4)\} \rangle$$
$$\mathcal{G}' = \langle V^{\mathcal{G}'}; E^{\mathcal{G}'} \rangle = \langle \{1,2,3,4,5\}; \{(1,2),(1,3),(1,4),(2,3),(3,4),(2,5),(3,5)\} \rangle$$

Oba grafy sestávají z množiny vrcholů, a z binární relace na této množině. Jde o struktury v *jazyce teorie grafů* $\mathcal{L} = \langle E \rangle$, kde E je binární *relační symbol. Jazyk* predikátové logiky specifikuje jaké relace (kolik a jakých arit — unární, binární, ternární, atd.) mají struktury mít, a jaké symboly pro ně budeme používat. Kromě toho používáme symbol rovnosti $=$ a struktury mohou obsahovat také funkce a konstanty (jako například funkce $+, -, \cdot$ a konstanty $0, 1$ v *tělese* reálných čísel), ty si ale necháme na později.

Predikátová logika používá tytéž logické spojky jako výroková logika, ale základním stavebním kamenem *predikátových formulí* nejsou výrokové proměnné, nýbrž tzv. *atomické formule*, například: $E(x, y)$ představuje tvrzení, že v grafu vede hrana z vrcholu x do vrcholu y. Zde x, y jsou *proměnné* reprezentující vrcholy daného grafu. Kromě toho ve formulích můžeme používat *kvantifikátory*: $(\forall x)$ "pro všechny vrcholy x" a $(\exists y)$ "existuje vrchol y".[6]

Nyní můžeme formalizovat tvrzení, která dávají smysl pro libovolný graf. Například:

- "V grafu nejsou smyčky":
$$(\forall x)(\neg E(x,x))$$
- "Existuje vrchol výstupního stupně 1":
$$(\exists x)(\exists y)(E(x,y) \wedge (\forall z)(E(x,z) \to y = z))$$

V daném grafu $\mathcal{G}$ a při dosazení vrcholu u za proměnnou x a vrcholu v za proměnnou y vyhodnotíme $E(x, y)$ jako True, právě když $(u, v) \in E^{\mathcal{G}}$.

[6] Kvantifikátory si můžeme představit jako "konjunkci" resp. "disjunkci" přes všechny vrcholy grafu.

1.2.3 Formalizace barvení grafů v predikátové logice

Vraťme se zpět k barvení grafů. Přirozený způsob jak formalizovat náš problém 3-obarvitelnosti je v jazyce $\mathcal{L}' = \langle E, R, G, B\rangle$, kde E je binární a R, G, B jsou unární relační symboly, tedy $R(x)$ znamená "vrchol x je červený". Strukturou pro tento jazyk je (orientovaný) graf spolu s trojicí množin vrcholů, např.

$$\begin{aligned}\mathcal{G}_C &= \langle V^{\mathcal{G}_C}; E^{\mathcal{G}_C}, R^{\mathcal{G}_C}, G^{\mathcal{G}_C}, B^{\mathcal{G}_C}\rangle \\ &= \langle \{1,2,3,4\}; \{(1,2),(1,3),(1,4),(2,3),(3,4)\}, \{1\}, \{2,4\}, \{3\}\rangle\end{aligned}$$

reprezentuje graf $\mathcal{G}$ s validním obarvením z obrázku výše. Budeme říkat, že $\mathcal{G}_C$ je *expanze* $\mathcal{L}$-struktury $\mathcal{G}$ do jazyka $\mathcal{L}'$.

Podobně jako ve výrokové logice musíme nejprve zajistit, aby naše modely reprezentovaly obarvené grafy. Začneme požadavkem, aby každý vrchol byl obarven nejvýše jednou barvou:

$$(\forall x)((\neg R(x) \vee \neg G(x)) \wedge (\neg R(x) \vee \neg B(x)) \wedge (\neg G(x) \vee \neg B(x)))$$

Obarvení alespoň jednou barvou vyjádříme takto:

$$(\forall x)(R(x) \vee G(x) \vee B(x))$$

A hranovou podmínku formalizujeme pomocí predikátu $E(x, y)$ například takto:

$$(\forall x)(\forall y)(E(x,y) \to ((\neg R(x) \vee \neg R(y)) \wedge (\neg G(x) \vee \neg G(y)) \wedge (\neg B(x) \vee \neg B(y))))$$

Modely takto vzniklé teorie reprezentují orientované grafy s vrcholovým 3-obarvením.

1.3 Další druhy logických systémů

Predikátové logice, kde proměnné reprezentují jednotlivé vrcholy, říkáme logika *prvního řádu* (anglicky *first-order (FO) logic*). V logice *druhého řádu* (anglicky *second-order (SO) logic*) máme také proměnné reprezentující množiny vrcholů nebo i množiny n-tic vrcholů (tj. relace, funkce). Například tvrzení, že daný graf je bipartitní, můžeme formalizovat v logice druhého řádu následující formulí, ve které S je *proměnná druhého řádu* reprezentující množinu vrcholů a $S(x)$ vyjadřuje, že "vrchol x je prvkem množiny S":

$$(\exists S)(\forall x)(\forall y)(E(x,y) \to (S(x) \leftrightarrow \neg S(y)))$$

Jako jiný, důležitý příklad uveďme tvrzení, že každá neprázdná zdola omezená podmnožina má infimum,[7] které můžeme formalizovat v logice druhého řádu takto:[8]

$$\begin{aligned}&(\forall S)((\exists x)S(x) \wedge (\exists x)(\forall y)(S(y) \to x \leq y) \to \\ &(\exists x)((\forall y)(S(y) \to x \leq y) \wedge (\forall z)((\forall y)(S(y) \to z \leq y) \to z \leq x)))\end{aligned}$$

A v logice *třetího řádu* máme i proměnné reprezentující množiny množin (což je užitečné např. v topologii), atd.

[7] Což platí v uspořádané množině reálných čísel, ale neplatí v racionálních číslech, např. $\{x \mid x^2 > 2, x > 0\}$.

[8] I když je tato formule velmi složitá, pokuste se porozumět jednotlivým částem.

Kromě výrokové a predikátové logiky existují i další typy logických systémů, například intuicionistická logika (která povoluje jen konstruktivní důkazy), temporální logiky (kde mluvíme o platnosti 'vždy', 'někdy v budoucnosti', 'dokud' apod.), modální logiky ('je možné', 'je nutné'), nebo fuzzy logiky (kde máme výroky '0.35 pravdivé'). Tyto logiky mají důležité aplikace v informatice, např. v umělé inteligenci (modální logiky pro uvažování autonomních agentů o svém okolí), v paralelním programování (temporální logiky), nebo v automatických pračkách (fuzzy logiky). V tomto kurzu se omezíme na výrokovou logiku a predikátovou logiku prvního řádu.

1.4 O přednášce

Přednáška je rozdělena na tři části:

První část se zabývá výrokovou logikou. Nejprve představíme syntaxi a sémantiku, dále problém splnitelnosti CNF formulí (známý NP-úplný problém SAT) a jeho polynomiálně řešitelné fragmenty (2-SAT, Horn-SAT). Ukážeme si také použití SAT solveru v praxi. Budeme pokračovat tablo metodu, u níž si dokážeme korektnost a úplnost, a také několik aplikací, například Větu u kompaktnosti. A na závěr představíme rezoluční metodu ve výrokové logice.

Ve druhé části představíme predikátovou logiku. Začneme opět syntaxí a sémantikou, ukážeme si jak lze adaptovat tablo metodu pro predikátovou logiku, a skončíme znovu rezoluční metodou. Kromě toho zmíníme několik praktických aplikací, např. databázové dotazy a logické programování (jazyk Prolog). Struktura výkladu v této části je silně provázaná s předchozí částí. Řada definic, vět, důkazů, i algoritmů bude velmi podobná svým protějškům ve výrokové logice. Lišit se budou převážně jen na nízké úrovni, v technických vnitřnostech. Proto je důležité dobře si uspořádat chápání výrokové logiky, než se pustíme do logiky predikátové.

Třetí, rozsahem nejmenší část je úvodem do teorie modelů, axiomatizovatelnosti, a algoritmické rozhodnutelnosti. Jde o ochutnávku pokročilejších témat, která lze potkat spíše v teoretické informatice a matematické logice, byť některá mají své místo i v informatice aplikované. Na závěr si představíme slavné Gödelovy věty o neúplnosti, které ukazují meze formální metody (formální dokazatelnosti v axiomatickém systému).

Část I

Výroková logika

Kapitola 2

Syntaxe a sémantika výrokové logiky

Syntaxe je soubor formálních pravidel pro tvoření korektních vět sestávajících ze slov (v případě přirozených jazyků), nebo formálních výrazů sestávajících ze symbolů (např. příkazy v programovacím jazyce). Naproti tomu *sémantika* popisuje význam takových výrazů. Vztah mezi syntaxí a sémantikou se prolíná celou logikou a je klíčem k jejímu pochopení.

2.1 Syntaxe výrokové logiky

Nejprve definujeme formální 'nápisy', se kterými budeme v logice pracovat.

2.1.1 Jazyk

Jazyk výrokové logiky je určený neprázdnou množinou *výrokových proměnných* $\mathbb{P}$ (také jim říkáme *prvovýroky* nebo *atomické výroky*). Tato množina může být konečná nebo i nekonečná, obvykle ale bude spočetná[1] (pokud neřekneme jinak), a bude mít dané uspořádání. Pro výrokové proměnné budeme obvykle používat označení p_i (od slova "proposition"), ale pro lepší čitelnost, zejména je-li $\mathbb{P}$ konečná, také $p, q, r, \ldots$ Například:

$$\mathbb{P}_1 = \{p, q, r\}$$
$$\mathbb{P}_2 = \{p_0, p_1, p_2, p_3, \ldots\} = \{p_i \mid i \in \mathbb{N}\}$$

Do jazyka patří kromě výrokových proměnných také *logické symboly*: symboly pro logické spojky $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ a závorky $(,)$. Budeme ale pro jednoduchost mluvit o "jazyce $\mathbb{P}$".

Poznámka 2.1.1. Pokud budeme potřebovat formálněji vyjádřit uspořádání jazyka $\mathbb{P}$, představíme si ho jako bijekci $\iota\colon \{0, 1, \ldots, n-1\} \to \mathbb{P}$ (pro konečný, n-prvkový jazyk) resp. $\iota\colon \mathbb{N} \to \mathbb{P}$ (je-li $\mathbb{P}$ spočetně nekonečný). V našich příkladech $\iota_1(0) = p$, $\iota_1(1) = q$, $\iota_1(2) = r$, a $\iota_2(i) = p_i$ pro všechna $i \in \mathbb{N}$.[2]

[1] To je důležité v mnoha aplikacích v informatice, nespočetné množiny se do (ani nekonečného) počítače nevejdou.

[2] Množina přirozených čísel $\mathbb{N}$ obsahuje nulu, viz standard ISO 80000-2:2019.

2.1.2 Výrok

Základním stavebním kamenem výrokové logiky je *výrok* neboli *výroková formule*. Je to konečný řetězec sestavený z výrokových proměnných a logických symbolů podle jistých pravidel. Prvovýroky jsou výroky, a dále můžeme vytvářet výroky z jednodušších výroků a logických symbolů: například pro logickou spojku $\wedge$ vypíšeme nejprve symbol '(', potom první výrok, symbol '$\wedge$', druhý výrok, a nakonec symbol ')'.

Definice 2.1.2 (Výrok)**.** *Výrok* (*výroková formule*) v jazyce $\mathbb{P}$ je prvek množiny $\mathrm{VF}_{\mathbb{P}}$ definované následovně: $\mathrm{VF}_{\mathbb{P}}$ je nejmenší množina splňující[3]

- pro každý prvovýrok $p \in \mathbb{P}$ platí $p \in \mathrm{VF}_{\mathbb{P}}$,
- pro každý výrok $\varphi \in \mathrm{VF}_{\mathbb{P}}$ je $(\neg\varphi)$ také prvek $\mathrm{VF}_{\mathbb{P}}$
- pro každé $\varphi, \psi \in \mathrm{VF}_{\mathbb{P}}$ jsou $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, a $(\varphi \leftrightarrow \psi)$ také prvky $\mathrm{VF}_{\mathbb{P}}$.

Výroky označujeme obvykle řeckými písmeny φ, ψ, χ (φ od slova "formule"). Abychom nemuseli vypisovat všechny čtyři binární logické spojky, používáme pro ně někdy zástupný symbol $\square$. Třetí bod definice bychom tedy mohli vyjádřit takto:

- pro každé $\varphi, \psi \in \mathrm{VF}_{\mathbb{P}}$ a $\square \in \{\wedge, \vee, \to, \leftrightarrow\}$ je $(\varphi \square \psi)$ také prvek $\mathrm{VF}_{\mathbb{P}}$.

Podvýrok (*podformule*) je podřetězec, který je sám o sobě výrokem. Uvědomte si, že všechny výroky jsou nutně konečné řetězce, vzniklé aplikací konečně mnoha kroků z definice na své podvýroky.

Příklad 2.1.3. Výrok $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \to (p \wedge q)))$ má následující podvýroky: $p, q, (\neg q), (p \vee (\neg q)), r, (p \wedge q), (r \to (p \wedge q)), \varphi$.

Výrok v jazyce $\mathbb{P}$ nemusí obsahovat všechny prvovýroky z $\mathbb{P}$ (ani nemůže pokud je $\mathbb{P}$ nekonečná množina). Bude se nám proto hodit značení $\mathrm{Var}(\varphi)$ pro množinu prvovýroků vyskytujících se ve φ.[4] V našem příkladě $\mathrm{Var}(\varphi) = \{p, q, r\}$.

Zavedeme si zkratky pro dva speciální výroky: $\top = (p \vee (\neg p))$ (*pravda*) a $\bot = (p \wedge (\neg p))$ (*spor*), kde $p \in \mathbb{P}$ je pevně zvolený (např. první prvovýrok z $\mathbb{P}$). Tedy výrok $\top$ je vždy pravdivý a výrok $\bot$ je vždy nepravdivý.

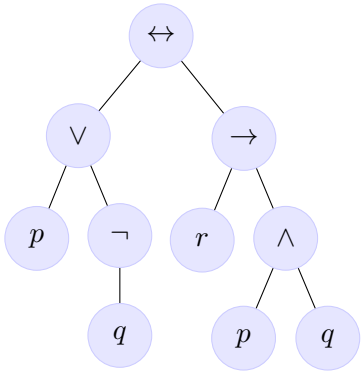
Při zápisu výroků můžeme pro lepší čitelnost některé závorky vynechat. Např. výrok φ z příkladu 2.1.3 můžeme reprezentovat nápisem $p \vee \neg q \leftrightarrow (r \to p \wedge q)$. Vynecháváme vnější závorky a používáme prioritu operátorů: $\neg$ má nejvyšší prioritu, dále $\wedge, \vee$, a konečně $\to, \leftrightarrow$ mají nejnižší prioritu. Dále nápisem $p \wedge q \wedge r \wedge s$ myslíme výrok $(p \wedge (q \wedge (r \wedge s)))$, a podobně pro $\vee$.[5][6]

[3] Takovému druhu definice říkáme *induktivní*. Lze také přirozeně vyjádřit pomocí *formální gramatiky*, viz předmět NTIN071 Automaty a gramatiky.

[4] Pokud nespecifikujeme v jakém jazyce je výrok (a pokud to není jasné z kontextu), myslíme tím, že je v jazyce $\mathrm{Var}(\varphi)$.

[5] Díky asociativitě $\wedge, \vee$ na uzávorkování nezáleží.

[6] Někdy se zavádí jemnější priority, $\wedge$ mívá vyšší prioritu než $\vee$, $\to$ vyšší než $\leftrightarrow$. A někdy se píše $p \to q \to r$ místo $(p \to (r \to q))$, byť $\to$ není asociativní a na uzávorkování záleží. Obojímu se ale raději vyhneme.



Obrázek 2.1: Strom výroku $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \rightarrow (p \wedge q)))$

2.1.3 Strom výroku

V definici výroku jsme zvolili *infixový* zápis se závorkami čistě z důvodu čitelnosti pro člověka. Nic by nám nebránilo použít *prefixový* zápis ("polskou notaci"), tj. definovat výroky takto:

- každý prvovýrok je výrok, a
- jsou-li φ, ψ výroky, jsou také $\neg\varphi$, $\wedge\varphi\psi$, $\vee\varphi\psi$, $\rightarrow\varphi\psi$, a $\leftrightarrow\varphi\psi$ výroky.

Výrok $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \rightarrow (p \wedge q)))$ bychom potom zapsali jako $\varphi = \leftrightarrow\vee p\neg q\rightarrow r\wedge pq$. Také bychom mohli použít *postfixový* zápis a psát $\varphi = pq\neg\vee rpq\wedge\rightarrow\leftrightarrow$. Vše podstatné o výroku ve skutečnosti obsahuje jeho stromová struktura, která zachycuje, jak je sestaven z jednodušších výroků, obdobně jako strom aritmetického výrazu.

Příklad 2.1.4. Strom výroku $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \rightarrow (p \wedge q)))$ je znázorněný na obrázku 2.1. Všimněte si také, že podvýroky φ odpovídají podstromům. Výrok φ získáme průchodem stromem od kořene, v každém vrcholu:

- pokud je label prvovýrok, vypíšeme ho
- pokud je label negace: vypíšeme '(¬', rekurzivně zavoláme syna, vypíšeme ')',
- jinak (pro binární logické spojky): vypíšeme '(', zavoláme levého syna, vypíšeme label, zavoláme pravého syna, vypíšeme ')'.[7]

Nyní si strom výroku definujeme formálně, *indukcí podle struktury výroku*:[8]

Definice 2.1.5 (Strom výroku)**.** *Strom výroku* φ, označme $\mathrm{Tree}(\varphi)$ je zakořeněný uspořádaný strom, definovaný induktivně takto:

- Je-li φ prvovýrok p, obsahuje $\mathrm{Tree}(\varphi)$ jediný vrchol, jeho label je p.
- Je-li φ tvaru $(\neg\varphi')$, má $\mathrm{Tree}(\varphi)$ kořen s labelem ¬, a jeho jediný syn je kořen $\mathrm{Tree}(\varphi')$.

[7] Prefixový a postfixový zápis bychom získali podobně, ale nevypisujeme závorky a label vypíšeme hned při vstupu resp. těsně před opuštěním vrcholu.

[8] Jakmile máme definovaný strom výroku, můžeme indukci podle struktury výroku chápat jako indukci podle hloubky stromu. Zatím tím ale chápejme indukci podle počtu kroků z definice 2.1.2, kterými výrok vznikl. Alternativně postačí indukce podle délky výroku, nebo podle počtu logických spojek.

- Je-li φ tvaru $(\varphi' \square \varphi'')$ pro $\square \in \{\wedge, \vee, \to, \leftrightarrow\}$, má $\mathrm{Tree}(\varphi)$ kořen s labelem $\square$ a dvěma syny: levý syn je kořen stromu $\mathrm{Tree}(\varphi')$, pravý je kořen $\mathrm{Tree}(\varphi'')$.

Cvičení 2.1. Dokažte, že každý výrok má jednoznačně určený strom výroku, a naopak.

2.1.4 Teorie

V praktických aplikacích nevyjádříme požadované vlastnosti jediným výrokem — to by musel být velmi dlouhý a složitý a špatně by se s ním pracovalo — ale mnoha jednoduššími výroky.

Definice 2.1.6 (Teorie)**.** *Teorie* v jazyce $\mathbb{P}$ je libovolná množina výroků v $\mathbb{P}$, tedy libovolná podmnožina $T \subseteq \mathrm{VF}_{\mathbb{P}}$. Jednotlivým výrokům $\varphi \in T$ říkáme také *axiomy*.

Konečné teorie by tedy bylo možné (byť ne praktické) nahradit jediným výrokem: konjunkcí všech jejich axiomů. Připouštíme ale i nekonečné teorie (triviálním příkladem je teorie $T = \mathrm{VF}_{\mathbb{P}}$), a prázdnou teorii $T = \emptyset$.[9]

2.2 Sémantika výrokové logiky

V naší logice je sémantika daná jednou ze dvou možných hodnot: *pravda*, nebo *nepravda*. (V jiných logických systémech může být sémantika zajímavější.)

2.2.1 Pravdivostní hodnota

Výrokům můžeme přiřadit jednu ze dvou možných pravdivostních hodnot: *pravdivý* (*True*, 1), nebo *lživý* (*False*, 0). Prvovýroky reprezentují jednoduchá, nadále nedělitelná tvrzení (proto jim také říkáme *atomické* výroky); pravdivostní hodnotu jim musíme přiřadit tak, aby odpovídala tomu, co chceme modelovat (proto jim říkáme *výrokové proměnné*). Jakmile ale *ohodnotíme* prvovýroky, pravdivostní hodnota libovolného složeného výroku je jednoznačně určená, a snadno ji spočteme podle stromu výroku:

Příklad 2.2.1. Spočtěme pravdivostní hodnotu výroku $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \to (p \wedge q)))$ při ohodnocení (a) $p = 0$, $q = 0$, $r = 0$, (b) $p = 1$, $q = 0$, $r = 1$. Postupujeme od listů směrem ke kořeni, podobně jako bychom vyhodnocovali např. aritmetický výraz. Výrok φ *platí* při ohodnocení z (a), *neplatí* při ohodnocení z (b). Viz obrázek 2.2.

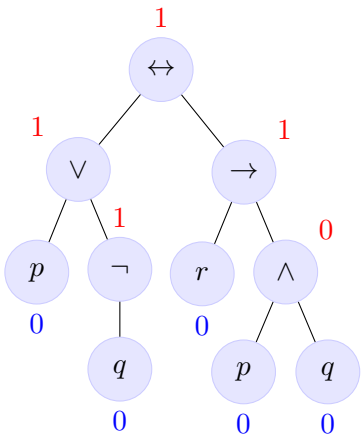
Logické spojky ve vnitřních vrcholech vyhodnocujeme podle jejich *pravdivostních tabulek*, viz tabulka 2.1.[10]

2.2.2 Výroky a booleovské funkce

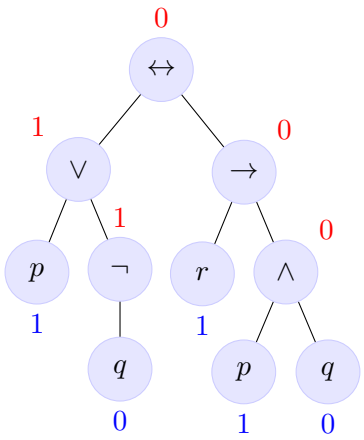
Abychom mohli formalizovat pravdivostní hodnotu výroku, podíváme se nejprve na souvislost výroků a booleovských funkcí.

Booleovská funkce je funkce $f\colon \{0,1\}^n \to \{0,1\}$, tedy vstupem je n-tice nul a jedniček, a výstupem 0 nebo 1. Každá logická spojka reprezentuje booleovskou funkci. V případě negace jde o unární funkci $f_\neg(x) = 1 - x$, ostatním logickým spojkám odpovídají binární funkce popsané v tabulce 2.2.

[9] Nekonečné teorie se hodí například pro popis vývoje nějakého systému v (diskrétním) čase $t = 0, 1, 2, \ldots$ Prázdná teorie se nehodí k ničemu, ale bylo by nešikovné formulovat věty o logice, pokud by teorie musely být



(a) $p = 0$, $q = 0$, $r = 0$



(b) $p = 1$, $q = 0$, $r = 1$

Obrázek 2.2: Pravdivostní ohodnocení výroku

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \to q$	$p \leftrightarrow q$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Tabulka 2.1: Pravdivostní tabulky logických spojek.

$f_\wedge(x,y)$:

	0	1
0	0	0
1	0	1

$f_\vee(x,y)$:

	0	1
0	0	1
1	1	1

$f_\to(x,y)$:

	0	1
0	1	1
1	0	1

$f_\leftrightarrow(x,y)$:

	0	1
0	1	0
1	0	1

Tabulka 2.2: Booleovské funkce logických spojek

Definice 2.2.2 (Pravdivostní funkce)**.** *Pravdivostní funkce* výroku φ v konečném jazyce $\mathbb{P}$ je funkce $f_{\varphi,\mathbb{P}}\colon \{0,1\}^{|\mathbb{P}|} \to \{0,1\}$ definovaná induktivně:

- je-li φ i-tý prvovýrok z $\mathbb{P}$, potom $f_{\varphi,\mathbb{P}}(x_0,\ldots,x_{n-1}) = x_i$,
- je-li $\varphi = (\neg\varphi')$, potom
$$f_{\varphi,\mathbb{P}}(x_0,\ldots,x_{n-1}) = f_{\neg}(f_{\varphi',\mathbb{P}}(x_0,\ldots,x_{n-1})),$$
- je-li $(\varphi' \square \varphi'')$ kde $\square \in \{\wedge,\vee,\to,\leftrightarrow\}$, potom
$$f_{\varphi,\mathbb{P}}(x_0,\ldots,x_{n-1}) = f_{\square}(f_{\varphi',\mathbb{P}}(x_0,\ldots,x_{n-1}), f_{\varphi'',\mathbb{P}}(x_0,\ldots,x_{n-1})).$$

Příklad 2.2.3. Spočtěme pravdivostní funkci výroku $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \to (p \wedge q)))$ v jazyce $\mathbb{P}' = \{p,q,r,s\}$:

$$f_{\varphi,\mathbb{P}'}(x_0,x_1,x_2,x_3) = f_{\leftrightarrow}(f_{\vee}(x_0, f_{\neg}(x_1)), f_{\to}(x_2, f_{\wedge}(x_0,x_1)))$$

Pravdivostní hodnotu výroku φ při ohodnocení $p=1$, $q=0$, $r=1$, $s=1$ spočteme takto (srovnejte s obrázkem 2.2(b)):

$$\begin{aligned} f_{\varphi,\mathbb{P}'}(1,0,1,1) &= f_{\leftrightarrow}(f_{\vee}(1, f_{\neg}(0)), f_{\to}(1, f_{\wedge}(1,0))) \\ &= f_{\leftrightarrow}(f_{\vee}(1,1), f_{\to}(1,0)) \\ &= f_{\leftrightarrow}(1,0) \\ &= 0 \end{aligned}$$

Pozorování 2.2.4. *Pravdivostní funkce výroku φ nad $\mathbb{P}$ závisí pouze na proměnných odpovídajících prvovýrokům z* $\mathrm{Var}(\varphi) \subseteq \mathbb{P}$.

Tedy i pokud máme výrok φ v *nekonečném* jazyce $\mathbb{P}$, můžeme se omezit na jazyk $\mathrm{Var}(\varphi)$ (který je konečný) a uvažovat pravdivostní funkci nad tímto jazykem.

2.2.3 Modely

Konkrétní pravdivostní ohodnocení výrokových proměnných představuje reprezentaci 'reálného světa' (systému) v námi zvoleném 'formálním světě', proto mu také říkáme *model*.

Definice 2.2.5 (Model jazyka)**.** *Model* jazyka $\mathbb{P}$ je libovolné pravdivostní ohodnocení $v\colon \mathbb{P} \to \{0,1\}$. *Množinu (všech) modelů jazyka* $\mathbb{P}$ označíme $\mathrm{M}_{\mathbb{P}}$:

$$\mathrm{M}_{\mathbb{P}} = \{v \mid v\colon \mathbb{P} \to \{0,1\}\} = \{0,1\}^{\mathbb{P}}$$

Modely budeme označovat písmeny v, u, w apod. (v od slova 'valuation'). Model jazyka je tedy funkce, formálně množina dvojic (vstup, výstup). Například pro jazyk $\mathbb{P} = \{p,q,r\}$ a pravdivostní ohodnocení ve kterém p je pravda, q nepravda, a r pravda máme model

$$v = \{(p,1),(q,0),(r,1)\}.$$

Pro jednoduchost ale budeme psát jen $v = (1,0,1)$. Pro jazyk $\mathbb{P} = \{p,q,r\}$ tedy máme $2^3 = 8$ modelů:

$$\mathrm{M}_{\mathbb{P}} = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$$

neprázdné.

[10] Připomeňme ještě jednou, že disjunkce není exkluzivní, tj. $p \vee q$ platí i pokud platí p i q, a že implikace je čistě logická, tj. $p \to q$ platí kdykoliv p neplatí.

Poznámka 2.2.6. Formálně vzato, ztotožňujeme množinu $\{0,1\}^{\mathbb{P}}$ s množinou $\{0,1\}^{|\mathbb{P}|}$ pomocí uspořádání ι jazyka $\mathbb{P}$ (viz Poznámka 2.1.1). Konkrétně, místo prvku $v = \{(p,1),(q,0),(r,1)\} \in \{0,1\}^{\mathbb{P}}$ píšeme $(1,0,1) = (v \circ \iota)(0,1,2) = (v(\iota(0)), v(\iota(1)), v(\iota(2))) \in \{0,1\}^{|\mathbb{P}|}$ (kde funkcím v, ι dovolíme působit 'po složkách').[11] Pokud by se to zdálo matoucí, představte si model v jako množinu prvovýroků, které jsou ohodnocené jako pravda, tj. $\{p,r\} \subseteq \mathbb{P}$, náš zápis $v = (1,0,1)$ je potom charakteristický vektor této množiny. Toto ztotožnění budeme nadále používat bez dalšího upozornění.

2.2.4 Platnost

Nyní můžeme definovat klíčový pojem logiky, *platnost* výroku v daném modelu. Neformálně, výrok platí v modelu (tj. při konkrétním pravdivostním ohodnocení prvovýroků), pokud jeho pravdivostní hodnota, tak jak jsme ji počítali v Příkladu 2.2.1, je rovna 1. Ve formální definici využijeme pravdivostní funkci výroku (Definice 2.2.2).[12]

Definice 2.2.7 (Platnost výroku v modelu, model výroku)**.** Mějme výrok φ v jazyce $\mathbb{P}$ a model $v \in \mathrm{M}_{\mathbb{P}}$. Pokud platí $f_{\varphi,\mathbb{P}}(v) = 1$, potom říkáme, že výrok φ *platí* v modelu v, v je *modelem* φ, a píšeme $v \models \varphi$. Množinu všech modelů výroku φ označujeme $\mathrm{M}_{\mathbb{P}}(\varphi)$.

Modelům jazyka, které nejsou modely φ, budeme někdy říkat *nemodely* φ. Tvoří doplněk množiny modelů φ. S pomocí standardního zápisu pro inverzní funkci můžeme psát:

$$\mathrm{M}_{\mathbb{P}}(\varphi) = \{v \in \mathrm{M}_{\mathbb{P}} \mid v \models \varphi\} = f^{-1}_{\varphi,\mathbb{P}}[1]$$
$$\overline{\mathrm{M}_{\mathbb{P}}(\varphi)} = M_{\mathbb{P}} \setminus M_{\mathbb{P}}(\varphi) = \{v \in \mathrm{M}_{\mathbb{P}} \mid v \not\models \varphi\} = f^{-1}_{\varphi,\mathbb{P}}[0]$$

Je-li jazyk zřejmý z kontextu, můžeme psát jen $\mathrm{M}(\varphi)$. Musíme si ale být opravdu jistí: například v jazyce $\mathbb{P} = \{p,q\}$ máme

$$\mathrm{M}_{\{p,q\}}(p \to q) = \{(0,0),(0,1),(1,1)\},$$

zatímco v jazyce $\mathbb{P}' = \{p,q,r\}$ bychom měli

$$\mathrm{M}_{\mathbb{P}'}(p \to q) = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,1,0),(1,1,1)\}.$$

Definice 2.2.8 (Platnost teorie, model teorie)**.** Je-li T teorie v jazyce $\mathbb{P}$, potom T *platí* v modelu v, pokud každý axiom $\varphi \in T$ platí ve v. V tom případě říkáme také, že v je *modelem* T, a píšeme $v \models T$. Množinu všech modelů teorie T v jazyce $\mathbb{P}$ označíme $\mathrm{M}_{\mathbb{P}}(T)$.

Pracujeme-li s konečnou teorií, nebo přidáváme-li k nějaké teorii konečně mnoho nových axiomů, budeme používat následující zjednodušený zápis:

- $\mathrm{M}_{\mathbb{P}}(\varphi_1, \varphi_2, \ldots, \varphi_n)$ místo $\mathrm{M}_{\mathbb{P}}(\{\varphi_1, \varphi_2, \ldots, \varphi_n\})$,
- $\mathrm{M}_{\mathbb{P}}(T, \varphi)$ místo $\mathrm{M}_{\mathbb{P}}(T \cup \{\varphi\})$.

Všimněte si, že $\mathrm{M}_{\mathbb{P}}(T,\varphi) = \mathrm{M}_{\mathbb{P}}(T) \cap \mathrm{M}_{\mathbb{P}}(\varphi)$, $\mathrm{M}_{\mathbb{P}}(T) = \bigcap_{\varphi \in T} \mathrm{M}_{\mathbb{P}}(\varphi)$, a že pro konečnou teorii (podobně i pro spočetnou) platí

$$\mathrm{M}_{\mathbb{P}}(\varphi_1) \supseteq \mathrm{M}_{\mathbb{P}}(\varphi_1,\varphi_2) \supseteq \mathrm{M}_{\mathbb{P}}(\varphi_1,\varphi_2,\varphi_3) \supseteq \cdots \supseteq \mathrm{M}_{\mathbb{P}}(\varphi_1,\varphi_2,\ldots,\varphi_n).$$

Toho můžeme využít při hledání modelů hrubou silou.

[11] Alternativně bychom mohli při formalizaci syntaxe vyžadovat (alespoň pro spočetné jazyky), aby jazyk *byl* $\mathbb{P} = \{0,1,2,\ldots\}$ a symboly p_0, p_1, p, q, r používat jen pro zvýšení čitelnosti.

[12] Pro *platnost* používáme symbol $\models$, který čteme jako 'splňuje' nebo 'modeluje', v LaTeXu \models.

Příklad 2.2.9. Modely teorie $T = \{p \vee q \vee r, q \to r, \neg r\}$ (v jazyce $\mathbb{P} = \{p, q, r\}$) můžeme najít tak, najdeme tak, že nejprve najdeme modely výroku $\neg r$:

$$\mathrm{M}_{\mathbb{P}}(r) = \{(x, y, 0) \mid x, y \in \{0, 1\}\} = \{(0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 0)\},$$

poté určíme, ve který z těchto modelů platí výrok $q \to r$:

- $(0, 0, 0) \models q \to r$,
- $(0, 1, 0) \not\models q \to r$,
- $(1, 0, 0) \models q \to r$,
- $(1, 1, 0) \not\models q \to r$,

Tedy $\mathrm{M}_{\mathbb{P}}(r, q \to r) = \{(0, 0, 0), (1, 0, 0)\}$. Výrok $p \vee q \vee r$ platí jen ve druhém z těchto modelů, dostáváme tedy

$$\mathrm{M}_{\mathbb{P}}(r, q \to r, p \vee q \vee r) = \mathrm{M}_{\mathbb{P}}(T) = \{(1, 0, 0)\}.$$

Tento postup je efektivnější než určit množiny modelů jednotlivých axiomů a udělat jejich průnik. (Ale mnohem méně efektivní než postup založený na tablo metodě, který si ukážeme později.)

2.2.5 Další sémantické pojmy

V návaznosti na pojem platnosti budeme používat řadu dalších pojmů. Pro některé vlastnosti existuje více různých termínů, v závislosti na kontextu v jakém se vyskytnou.

Definice 2.2.10 (Sémantické pojmy)**.** Říkáme, že výrok φ (v jazyce $\mathbb{P}$) je

- *pravdivý*, *tautologie*, *platí (v logice/logicky)*, a píšeme $\models \varphi$, pokud platí v každém modelu (jazyka $\mathbb{P}$), $\mathrm{M}_{\mathbb{P}}(\varphi) = \mathrm{M}_{\mathbb{P}}$,
- *lživý*, *sporný*, pokud nemá žádný model, $\mathrm{M}_{\mathbb{P}}(\varphi) = \emptyset$.[13]
- *nezávislý*, pokud platí v nějakém modelu, a neplatí v nějakém jiném modelu, tj. není pravdivý ani lživý, $\emptyset \subsetneq \mathrm{M}_{\mathbb{P}}(\varphi) \subsetneq \mathrm{M}_{\mathbb{P}}$,
- *splnitelný*, pokud má nějaký model, tj. není lživý, $\mathrm{M}_{\mathbb{P}}(\varphi) \neq \emptyset$.

Dále říkáme, že výroky φ, ψ (ve stejném jazyce $\mathbb{P}$) jsou *(logicky) ekvivalentní*, píšeme $\varphi \sim \psi$ pokud mají stejné modely, tj.

$$\varphi \sim \psi \text{ právě když } \mathrm{M}_{\mathbb{P}}(\varphi) = \mathrm{M}_{\mathbb{P}}(\psi).$$

Příklad 2.2.11. Například platí následující:

- výroky $\top$, $p \vee q \leftrightarrow q \vee p$ jsou pravdivé,
- výroky $\bot$, $(p \vee q) \wedge (p \vee \neg q) \wedge \neg p$ jsou lživé,
- výroky $p, p \wedge q$ jsou nezávislé, a také splnitelné, a

[13] Všimněte si, že být *lživý* není totéž, co nebýt *pravdivý*!

- následující výroky jsou ekvivalentní:
  - $p \sim p \vee p \sim p \vee p \vee p$,
  - $p \to q \sim \neg p \vee q$,
  - $\neg p \to (p \to q) \sim \top$.

Pojmy z Definice 2.2.10 můžeme také relativizovat vzhledem k dané teorii. To znamená, že se v jednotlivých definicích omezíme na modely této teorie:

Definice 2.2.12 (Sémantické pojmy vzhledem k teorii)**.** Mějme teorii T v jazyce $\mathbb{P}$. Říkáme, že výrok φ v jazyce $\mathbb{P}$ je

- *pravdivý v T*, *důsledek T*, *platí v T*, a píšeme $T \models \varphi$, pokud φ platí v každém modelu teorie T, neboli $\mathrm{M}_{\mathbb{P}}(T) \subseteq \mathrm{M}_{\mathbb{P}}(\varphi)$,
- *lživý v T*, *sporný v T*, pokud neplatí v žádném modelu T, neboli $\mathrm{M}_{\mathbb{P}}(\varphi) \cap \mathrm{M}_{\mathbb{P}}(T) = \mathrm{M}_{\mathbb{P}}(T, \varphi) = \emptyset$.
- *nezávislý v T*, pokud platí v nějakém modelu T, a neplatí v nějakém jiném modelu T, tj. není pravdivý v T ani lživý v T, $\emptyset \subsetneq \mathrm{M}_{\mathbb{P}}(T, \varphi) \subsetneq \mathrm{M}_{\mathbb{P}}(T)$,
- *splnitelný v T*, *konzistentní s T*, pokud platí v nějakém modelu T, tj. není lživý v T, $\mathrm{M}_{\mathbb{P}}(T, \varphi) \neq \emptyset$.

A říkáme, že výroky φ, ψ (ve stejném jazyce $\mathbb{P}$) jsou *ekvivalentní v T*, *T-ekvivalentní*, píšeme $\varphi \sim_T \psi$ pokud platí v týchž modelech T, tj.

$$\varphi \sim_T \psi \text{ právě když } \mathrm{M}_{\mathbb{P}}(T, \varphi) = \mathrm{M}_{\mathbb{P}}(T, \psi).$$

Všimněte si, že pro prázdnou teorii $T = \emptyset$ platí $\mathrm{M}_{\mathbb{P}}(T) = \mathrm{M}_{\mathbb{P}}$ a výše uvedené pojmy pro T se proto shodují s původními. Opět si pojmy ilustrujeme na několika příkladech:

Příklad 2.2.13. Mějme teorii $T = \{p \vee q, \neg r\}$. Platí následující:

- výroky $q \vee p$, $\neg p \vee \neg q \vee \neg r$ jsou pravdivé v T,
- výrok $(\neg p \wedge \neg q) \vee r$ je v T lživý,
- výroky $p \leftrightarrow q, p \wedge q$ jsou v T nezávislé, a také splnitelné, a
- platí $p \sim_T p \vee r$ (ale $p \not\sim p \vee r$).

2.2.6 Univerzálnost logických spojek

V jazyce výrokové logiky používáme následující logické spojky: $\neg, \wedge, \vee, \to, \leftrightarrow$. To ale není jediná možná volba, k vybudování plnohodnotné logiky by nám stačila například negace a implikace,[14] nebo negace, konjunkce, a disjunkce.[15] A jak uvidíme níže, mohli bychom použít i jiné logické spojky. Naše volba je zlatou střední cestou mezi bohatostí vyjadřování na jedné straně, a úsporností syntaktických pravidel na straně druhé.

[14] Negaci potřebujeme k popisu stavu systému, a implikaci k popisu chování v čase.
[15] Ty stačí k vybudování logických obvodů.

Co myslíme tím, že je logika plnohodnotná? Řekneme, že množina logických spojek S je *univerzální*, pokud lze každou booleovskou funkci f vyjádřit jako pravdivostní funkci $f_{\varphi,\mathbb{P}}$ nějakého výroku φ vybudovaného z logických spojek z S (kde $|\mathbb{P}| = n$ je-li f n-ární funkce). Ekvivalentně, pro každý konečný jazyk $\mathbb{P}$ (řekněme, že n-prvkový) a každou množinu modelů $M \subseteq \mathrm{M}_{\mathbb{P}}$ musí existovat výrok φ takový, že $\mathrm{M}_{\mathbb{P}}(\varphi) = M$. (Ekvivalence těchto dvou vyjádření plyne z toho, že máme-li booleovskou funkci f a zvolíme-li $M = f^{-1}[1]$, potom $f_{\varphi,\mathbb{P}} = f$ právě když $\mathrm{M}_{\mathbb{P}}(\varphi) = M$.)

Tvrzení 2.2.14. *Množiny logických spojek* $\{\neg, \wedge, \vee\}$ *a* $\{\neg, \to\}$ *jsou univerzální.*

Důkaz. Mějme funkci $f\colon \{0,1\}^n \to \{0,1\}$, resp. množinu modelů $M = f^{-1}[1] \subseteq \{0,1\}^n$. Náš jazyk bude $\mathbb{P} = \{p_1, \ldots, p_n\}$. Pokud by množina M obsahovala jediný model, např. $v = (1,0,1,0)$ mohli bychom ji reprezentovat výrokem $\varphi_v = p_1 \wedge \neg p_2 \wedge p_3 \wedge \neg p_4$, který říká 'musím být model v'. Pro obecný model v bychom výrok φ_v zapsali takto:

$$\varphi_v = p_1^{v_1} \wedge p_2^{v_2} \wedge \cdots \wedge p_n^{v_n} = \bigwedge_{i=1}^{n} p_i^{v(p_i)} = \bigwedge_{p\in\mathbb{P}} p^{v(p)}$$

kde zavádíme následující užitečné značení: $p^{v(p)}$ je výrok p pokud $v(p) = 1$, a výrok $\neg p$ pokud $v(p) = 0$.

Obsahuje-li množina M více modelů, řekneme 'musím být alespoň jeden z modelů z M':

$$\varphi_M = \bigvee_{v\in M} \varphi_v = \bigvee_{v\in M} \bigwedge_{p\in\mathbb{P}} p^{v(p)}$$

Zřejmě platí $M_{\mathbb{P}}(\varphi_M) = M$ neboli $f_{\varphi_M,\mathbb{P}} = f$. (Pokud $M = \emptyset$, potom z definice $\bigvee_{v\in M} \varphi_v = \bot$.)[16]

Univerzálnost $\{\neg, \to\}$ plyne z univerzálnosti $\{\neg, \wedge, \vee\}$ a faktu, že konjunkci a disjunkci můžeme vyjádřit pomocí negace a implikace: $p \wedge q \sim \neg(p \to \neg q)$ a $p \vee q \sim \neg p \to q$. □

Poznámka 2.2.15. Všimněte si, že při konstrukci výroku φ_M je klíčové, že množina M je konečná (má nejvýše 2^n prvků). Kdyby byla nekonečná, symbol '$\bigvee_{v\in M}$' by znamenal 'disjunkci' nekonečně mnoha výroků, a výsledkem by tedy nebyl konečný nápis, tj. 'φ_M' by vůbec nebyl výrok. (Máme-li spočetně nekonečný jazyk $\mathbb{P}'$, potom ne každou podmnožinu $M \subseteq \mathrm{M}_{\mathbb{P}'}$ lze reprezentovat výrokem—takových podmnožin je nespočetně mnoho, zatímco výroků je jen spočetně mnoho.)

Jaké další logické spojky bychom mohli použít? Nulární booleovské funkce,[17] neboli konstanty 0, 1, bychom mohli zavést jako symboly TRUE a FALSE, my si ale vystačíme s výroky $\top, \bot$. Unární booleovské funkce jsou čtyři ($4 = 2^{2^1}$), ale negace je jediná 'zajímavá': ostatní jsou $f(x) = x$, $f(x) = 0$, a $f(x) = 1$. Zajímavých binárních logických spojek už je více, v přírodě se vyskytují například tyto:

- NAND neboli *Shefferova spojka*, někdy se používá symbol $p \uparrow q$, platí $p \uparrow q \sim \neg(p \wedge q)$,
- NOR neboli *Pierceova spojka*, někdy se používá symbol $p \downarrow q$, platí $p \downarrow q \sim \neg(p \vee q)$,

[16] Podobně jako součet prázdné množiny sčítanců je roven 0.

[17] Ve formalizaci matematiky resp. informatiky funkce arity 0 znamená, že nemá žádné vstupy, výstup tedy nemůže záviset na vstupu a je konstantní. Formálně, jde o funkce $f\colon \emptyset \to \{0,1\}$. Pokud je to matoucí, představte si, že funkce musí mít aritu alespoň 1, a místo 'nulární funkce' říkejme 'konstanta'.

- XOR, neboli *exclusive-OR*, někdy se píše také $\oplus$, platí $p \oplus q \sim (p \vee q) \wedge \neg(p \wedge q)$, neboli součet pravdivostní hodnot modulo 2.

Cvičení 2.2. Vyjádřete $(p \oplus q) \oplus r$ pomocí $\{\neg, \wedge, \vee\}$.

Cvičení 2.3. Ukažte, že {NAND} a také {NOR} jsou univerzální.

Cvičení 2.4. Uvažme ternární logickou spojku IFTE, kde $IFTE(p, q, r)$ je splněno, právě když platí 'if p then q else r'. Určete pravdivostní tabulku této logické spojky (tj. funkci f_{IFTE}) a ukažte, že $\{\text{TRUE}, \text{FALSE}, \text{IFTE}\}$ je univerzální.

2.3 Normální formy

Připomeňme, že výroky jsou ekvivalentní, pokud mají stejnou množinu modelů. Pro každý výrok existuje nekonečně mnoho ekvivalentních výroků; často se hodí vyjádřit výrok v nějakém 'hezkém' (užitečném) 'tvaru', tj. najít ekvivalentní výrok v daném tvaru. Takovému konceptu tvaru se v matematice říká *normální forma*. My si představíme dvě nejznámější: *konjunktivní normální formu (conjunctive normal form, CNF)* a *disjunktivní normální formu (DNF)*.

Používá se následující terminologie a značení:

- *Literál* ℓ je buď prvovýrok p nebo negace prvovýroku $\neg p$. Pro prvovýrok p označme $p^0 = \neg p$ a $p^1 = p$. Je-li ℓ literál, potom $\bar{\ell}$ označuje *opačný literál* k ℓ. Je-li $\ell = p$ (*pozitivní literál*), potom $\bar{\ell} = \neg p$, je-li $\ell = \neg p$ (*negativní literál*), potom $\bar{\ell} = p$
- *Klauzule (clause)* je disjunkce literálů $C = \ell_1 \vee \ell_2 \vee \cdots \vee \ell_n$. *Jednotková klauzule (unit clause)* je samotný literál ($n = 1$) a *prázdnou klauzulí* ($n = 0$) myslíme $\bot$.
- Výrok je *v konjunktivní normální formě (v CNF)* pokud je konjunkcí klauzulí. *Prázdný výrok v CNF* je $\top$.
- *Elementární konjunkce* je konjunkce literálů $E = \ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n$. *Jednotková elementární konjunkce* je samotný literál ($n = 1$). *Prázdná elementární konjunkce* ($n = 0$) je $\top$.
- Výrok je *v disjunktivní normální formě (v DNF)* pokud je disjunkcí elementárních konjunkcí. *Prázdný výrok v DNF* je $\bot$.

Příklad 2.3.1. Výrok $p \vee q \vee \neg r$ je v CNF (je to jediná klauzule) a zároveň v DNF (je to disjunkce jednotkových elementárních konjunkcí). Výrok $(p \vee q) \wedge (p \vee \neg q) \wedge \neg p$ je v CNF, výrok $\neg p \vee (p \wedge q)$ je v DNF.

Příklad 2.3.2. Výrok φ_v z důkazu Tvrzení 2.2.14 je v CNF (je to konjunkce jednotkových klauzulí, tj. literálů) a také v DNF (je to jediná elementární konjunkce). Výrok φ_M je v DNF.

Pozorování 2.3.3. *Všimněte si, že výrok v CNF je tautologie, právě když každá jeho klauzule obsahuje dvojici opačných literálů. Podobně, výrok v DNF je splnitelný, pokud ne každá elementární konjunkce obsahuje dvojici opačných literálů.*

2.3.1 O dualitě

Všimněte si, že pokud ve výrokové logice zaměníme hodnoty pro pravdu a nepravdu, tj. 0 a 1, pravdivostní tabulka negace zůstává stejná, z konjunkce se stává disjunkce, a naopak. Tomuto konceptu se říká *dualita*; v logice uvidíme mnoho příkladů.

Platí $\neg(p \wedge q) \sim (\neg p \vee \neg q)$ a *z duality* víme také $\neg(\neg p \vee \neg q) \sim (\neg\neg p \wedge \neg\neg q)$, z čehož snadno odvodíme $\neg(p \vee q) \sim (\neg p \wedge \neg q)$.[18] Obecněji, n-ární booleovské funkce f, g jsou navzájem *duální*, pokud platí pokud $f(\neg x) = \neg g(x)$. Máme-li výrok φ vybudovaný z $\{\neg, \wedge, \vee\}$ a zaměníme-li v něm $\wedge$ a $\vee$, a znegujeme-li výrokové proměnné (resp. zaměníme-li literály za opačné literály), dostáváme výrok $\psi \sim \neg\varphi$ (tj. modely φ jsou nemodely ψ a naopak), a funkce $f_{\varphi,\mathbb{P}}, f_{\psi,\mathbb{P}}$ jsou navzájem duální.

Pojem DNF je duální k pojmu CNF, 'je tautologie' je duální k 'není splnitelný', předchozí pozorování tedy můžeme chápat jako příklad duality. Ke každému tvrzení ve výrokové logice získáváme 'zdarma' tvrzení *duální*, vzniklé záměnou $\wedge$ a $\vee$, pravdy a nepravdy.

2.3.2 Převod do normální formy

Disjunktivní normální formu jsme již potkali, v důkazu Tvrzení 2.2.14. Klíčovou část důkazu bychom mohli zformulovat takto: 'Je-li jazyk konečný, lze každou množinu modelů *axiomatizovat* výrokem v DNF'. Z duality dostáváme také axiomatizaci v CNF, neboť doplněk množiny modelů je také množina modelů:

Tvrzení 2.3.4. *Mějme konečný jazyk $\mathbb{P}$ a libovolnou množinu modelů $M \subseteq \mathrm{M}_{\mathbb{P}}$. Potom existuje výrok φ_{DNF} v DNF a výrok φ_{CNF} v CNF takový, že $M = \mathrm{M}_{\mathbb{P}}(\varphi_{\mathrm{DNF}}) = \mathrm{M}_{\mathbb{P}}(\varphi_{\mathrm{CNF}})$. Konkrétně:*

$$\varphi_{\mathrm{DNF}} = \bigvee_{v \in M} \bigwedge_{p \in \mathbb{P}} p^{v(p)}$$
$$\varphi_{\mathrm{CNF}} = \bigwedge_{v \in \overline{M}} \bigvee_{p \in \mathbb{P}} \overline{p^{v(p)}} = \bigwedge_{v \notin M} \bigvee_{p \in \mathbb{P}} p^{1-v(p)}$$

Důkaz. Pro výrok φ_{DNF} viz důkaz Tvrzení 2.2.14, každá elementární konjunkce popisuje jeden model. Výrok φ_{CNF} je duální k výroku φ'_{DNF} sestrojenému pro doplněk $M' = \overline{M}$. Nebo můžeme dokázat přímo: modely klauzule $C_v = \bigvee_{p \in \mathbb{P}} p^{1-v(p)}$ jsou všechny modely kromě v, $\mathrm{M}_C = \mathrm{M}_{\mathbb{P}} \setminus \{v\}$, tedy každá klauzule v konjunkci zakazuje jeden nemodel. □

Tvrzení 2.3.4 dává návod, jak převádět výrok do disjunktivní nebo do konjunktivní normální formy:

Příklad 2.3.5. Uvažme výrok $\varphi = p \leftrightarrow (q \vee \neg r)$. Nejprve najdeme množinu modelů: $M = \mathrm{M}(\varphi) = \{(0,0,1), (1,0,0), (1,1,0), (1,1,1)\}$. Nyní najdeme výroky $\varphi_{\mathrm{DNF}}, \varphi_{\mathrm{CNF}}$ podle Tvrzení 2.3.4, ty mají stejnou množinu modelů jako φ, jsou tedy ekvivalentní.

Výrok φ_{DNF} najdeme tak, že pro každý model sestrojíme elementární konjunkci vynucující právě tento model:

$$\varphi_{\mathrm{DNF}} = (\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r)$$

Při konstrukci φ_{CNF} budeme potřebovat *nemodely* φ, $\overline{M} = \{(0,0,0), (0,1,0), (0,1,1), (1,0,1)\}$. Každá klauzule zakáže jeden nemodel:

$$\varphi_{\mathrm{CNF}} = (p \vee q \vee r) \wedge (p \vee \neg q \vee r) \wedge (p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee \neg r)$$

[18] Protože p, q jsou výrokové proměnné, mohou za ně být dosazeny obě hodnoty 0 i 1, tedy je můžeme zaměnit za k nim opačné literály.

Důsledek 2.3.6. *Každý výrok (v libovolném, i nekonečném jazyce $\mathbb{P}$) je ekvivalentní nějakému výroku v CNF a nějakému výroku v DNF.*

Důkaz. I když je jazyk $\mathbb{P}$ nekonečný, výrok φ obsahuje jen konečně mnoho výrokových proměnných, můžeme tedy použít Tvrzení 2.3.4 pro jazyk $\mathbb{P}' = \mathrm{Var}(\varphi)$, a množinu modelů $M = \mathrm{M}_{\mathbb{P}'}(\varphi)$. Protože $M = \mathrm{M}_{\mathbb{P}'}(\varphi_{\mathrm{DNF}}) = \mathrm{M}_{\mathbb{P}'}(\varphi_{\mathrm{CNF}})$, máme $\varphi \sim \varphi_{\mathrm{DNF}} \sim \varphi_{\mathrm{CNF}}$. □

Cvičení 2.5. Rozmyslete si, jak lze z DNF výroku snadno vygenerovat jeho modely, a z CNF výroku jeho nemodely.

Poznámka 2.3.7. Kdy lze axiomatizovat *teorii* výrokem v DNF nebo výrokem v CNF? Mějme jazyk $\mathbb{P}' = \mathrm{Var}(T)$ (tj. všechny výrokové proměnné vyskytující se v axiomech T). Má-li T v jazyce $\mathbb{P}'$ konečně mnoho modelů (tj. je-li $\mathrm{M}_{\mathbb{P}'}(T)$ konečná), můžeme sestrojit výrok v DNF, a má-li konečně mnoho *nemodelů*, můžeme sestrojit výrok v CNF. Obecně ale ne každou teorii lze axiomatizovat *jediným* výrokem v CNF nebo v DNF. Vždy můžeme převést jednotlivé axiomy do CNF (nebo DNF), a můžeme také axiomatizovat teorii jen pomocí (potenciálně nekonečně mnoha) klauzulí.

Tento způsob převodu do CNF resp. do DNF vyžaduje znalost množiny modelů výroku, je tedy poměrně neefektivní. A také výsledná normální forma může být velmi dlouhá. Ukážeme si ještě jeden postup.

Převod pomocí ekvivalentních úprav

Využijeme následujícího pozorování: Nahradíme-li nějaký podvýrok ψ výroku φ ekvivalentním výrokem ψ', výsledný výrok φ' bude také ekvivalentní φ. Nejprve si ukážeme postup na příkladě:

Příklad 2.3.8. Převedeme opět výrok $\varphi = p \leftrightarrow (q \vee \neg r)$. Nejprve se zbavíme ekvivalence, vyjádříme ji jako konjunkci dvou implikací. V dalším kroku odstraníme implikace, pomocí pravidla $\varphi \to \psi \sim \neg\varphi \vee \psi$:

$$\begin{aligned} p \leftrightarrow (q \vee \neg r) &\sim (p \to (q \vee \neg r)) \wedge ((q \vee \neg r) \to p) \\ &\sim (\neg p \vee q \vee \neg r) \wedge (\neg(q \vee \neg r) \vee p) \end{aligned}$$

Nyní si představme strom výroku, v dalším kroku chceme dostat negace na co nejnižší úroveň stromu, bezprostředně nad listy: využijeme toho, že $\neg(q \vee \neg r) \sim \neg q \wedge \neg\neg r$ a zbavíme se dvojité negace $\neg\neg r \sim r$. Dostáváme výrok

$$(\neg p \vee q \vee \neg r) \wedge ((\neg q \wedge r) \vee p)$$

Nyní již necháme literály nedotčené, a použijeme distributivitu $\wedge$ vůči $\vee$, nebo naopak, podle toho, zda chceme DNF nebo CNF. Pro převod do CNF použijeme úpravu $(\neg q \wedge r) \vee p \sim (\neg q \vee p) \wedge (r \vee p)$, kterou jsme dostali symbol $\vee$ na nižší úroveň stromu. (Nakreslete si!) Tím už dostáváme výrok v CNF, pro přehlednost ještě seřadíme literály v klauzulích:

$$(\neg p \vee q \vee \neg r) \wedge (p \vee \neg q) \wedge (p \vee r)$$

Při převodu do DNF bychom postupovali obdobně, opakovanou aplikací distributivity. Zde vyjdeme z CNF formy a zkombinujeme každý literál z první klauzule s každým literálem z druhé a s každým literálem z třetí klauzule. Všimneme si, že stejný literál nemusíme v elementární konjunkci opakovat dvakrát, a že obsahuje-li elementární klauzule dvojici opačných

literálů, je sporná, a můžeme ji tedy v DNF vynechat. Také můžeme vynechat elementární konjunkci E, pokud máme jinou elementární konjunkci E' takovou, že E' obsahuje všechny literály obsažené v E, např. $E = (p \wedge \neg r)$ a $E' = (p \wedge q \wedge \neg r)$. (Rozmyslete si proč, a zformulujte duální zjednodušení při převodu do CNF.) Výsledný výrok v DNF je:

$$(\neg p \wedge \neg q \wedge r) \vee (p \wedge q \wedge r) \vee (p \wedge \neg r)$$

Nyní vypíšeme všechny potřebné ekvivalentní úpravy. Důkaz, že každý výrok lze převést do DNF a do CNF lze snadno provést indukcí podle struktury výroku (podle hloubky stromu výroku).

- Implikace a ekvivalence:

  $$\varphi \to \psi \sim \neg\varphi \vee \psi$$
  $$\varphi \leftrightarrow \psi \sim (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi)$$

- Negace:

  $$\neg(\varphi \wedge \psi) \sim \neg\varphi \vee \neg\psi$$
  $$\neg(\varphi \vee \psi) \sim \neg\varphi \wedge \neg\psi$$
  $$\neg\neg\varphi \sim \varphi$$

- Konjunkce (převod do DNF):

  $$\varphi \wedge (\psi \vee \chi) \sim (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$$
  $$(\varphi \vee \psi) \wedge \chi \sim (\varphi \wedge \chi) \vee (\psi \wedge \chi)$$

- Disjunkce (převod do CNF):

  $$\varphi \vee (\psi \wedge \chi) \sim (\varphi \vee \psi) \wedge (\varphi \vee \chi)$$
  $$(\varphi \wedge \psi) \vee \chi \sim (\varphi \vee \chi) \wedge (\psi \vee \chi)$$

Jak uvidíme v příští kapitole, CNF je v praxi mnohem důležitější než DNF (byť jde o duální pojmy). Při popisu reálného systému je přirozenější vyjádření pomocí konjunkce mnoha jednodušších vlastností, než jako jednu velmi dlouhou disjunkci. Existuje mnoho dalších forem reprezentace booleovských funkcí. Podobně jako datové struktury, vhodnou formu reprezentace volíme podle toho, jaké operace potřebujeme s funkcí dělat.[19]

2.4 Vlastnosti a důsledky teorií

Podívejme se nyní hlouběji na vlastnosti teorií. Podobně jako pro výroky řekneme, že dvě teorie T, T' v jazyce $\mathbb{P}$ jsou *ekvivalentní*, pokud mají stejnou množinu modelů:

$$T \sim T' \text{ právě když } \mathrm{M}_{\mathbb{P}}(T) = \mathrm{M}_{\mathbb{P}}(T')$$

Jde tedy o teorie vyjadřující tytéž vlastnosti modelů, jen jinak vyjádřené (*axiomatizované*). V logice nás převážně zajímají ty vlastnosti teorií, které nezávisí na konkrétní *axiomatizaci*.

Příklad 2.4.1. Například teorie $T = \{p \to q, p \leftrightarrow r\}$ je ekvivalentní teorii $T' = \{(\neg p \vee q) \wedge (\neg p \vee r) \wedge (p \vee \neg r)\}$.

Definice 2.4.2 (Vlastnosti teorií)**.** Řekneme, že teorie T v jazyce $\mathbb{P}$ je

- *sporná*, jestliže v ní platí $\bot$ (spor), ekvivalentně, jestliže nemá žádný model, ekvivalentně, jestliže v ní platí všechny výroky,
- *bezesporná* (*splnitelná*), pokud není sporná, tj. má nějaký model,

[19] Viz například přednáška NAIL031 Reprezentace booleovských funkcí.

- *kompletní*, jestliže není sporná a každý výrok je v ní pravdivý nebo lživý (tj. nemá žádné nezávislé výroky), ekvivalentně, pokud má právě jeden model.

Rozmysleme si, proč platí ekvivalence vlastností v definici. Uvědomme si, že ve sporné teorii platí skutečně platí všechny výroky! Vskutku, výrok platí v T, pokud platí v každém modelu T, ty ale žádné nejsou. Naopak, pokud teorie má alespoň jeden model, v tomto modelu nemůže platit $\bot = p \wedge \neg p$.

A je-li teorie kompletní, nemůže mít dva různé modely $v \neq v'$. Výrok $\varphi_v = \bigwedge_{p \in \mathbb{P}} p^{v(p)}$ (který jsme potkali v důkazu Tvrzení 2.2.14) by totiž byl nezávislý v T, protože platí v modelu v ale ne v modelu v'. Naopak, má-li T jediný model v, potom každý výrok buď platí ve v, a tedy platí v T, nebo neplatí ve v a potom je lživý v T.

Příklad 2.4.3. Příkladem sporné teorie je třeba $T_1 = \{p, p \to q, \neg q\}$. Teorie $T_2 = \{p \vee q, r\}$ je bezesporná, ale není kompletní, například výrok $p \wedge q$ v ní není pravdivý (neplatí v modelu $(1, 0, 1)$) ale ani lživý (platí v modelu $(1, 1, 1)$). Teorie $T_2 \cup \{\neg p\}$ je kompletní, jejím jediným modelem je $(0, 1, 1)$.

2.4.1 Důsledky teorií

Připomeňme, že důsledek teorie T je každý výrok, který v T platí (tj. platí v každém modelu T) a označme si *množinu všech důsledků* teorie T v jazyce $\mathbb{P}$ jako

$$\mathrm{Csq}_{\mathbb{P}}(T) = \{\varphi \in \mathrm{VF}_{\mathbb{P}} \mid T \models \varphi\}$$

Pokud je teorie T v jazyce $\mathbb{P}$, můžeme psát:

$$\mathrm{Csq}_{\mathbb{P}}(T) = \{\varphi \in \mathrm{VF}_{\mathbb{P}} \mid \mathrm{M}_{\mathbb{P}}(T) \subseteq \mathrm{M}_{\mathbb{P}}(\varphi)\}$$

(Dává ale smysl mluvit i o důsledcích teorie v nějakém menším jazyce, který je podmnožinou jazyka T).

Ukážeme si několik jednoduchých vlastností důsledků:

Tvrzení 2.4.4. *Mějme teorie T, T' a výroky $\varphi, \varphi_1, \ldots, \varphi_n$ v jazyce $\mathbb{P}$. Potom platí:*

(i) $T \subseteq \mathrm{Csq}_{\mathbb{P}}(T)$,

(ii) $\mathrm{Csq}_{\mathbb{P}}(T) = \mathrm{Csq}_{\mathbb{P}}(\mathrm{Csq}_{\mathbb{P}}(T))$,

(iii) pokud $T \subseteq T'$, *potom* $\mathrm{Csq}_{\mathbb{P}}(T) \subseteq \mathrm{Csq}_{\mathbb{P}}(T')$,

(iv) $\varphi \in \mathrm{Csq}_{\mathbb{P}}(\{\varphi_1, \ldots, \varphi_n\})$ *právě když je výrok* $(\varphi_1 \wedge \cdots \wedge \varphi_n) \to \varphi$ *tautologie.*

Důkaz. Důkaz je snadný, použijeme-li, že φ je důsledek T právě když $\mathrm{M}_{\mathbb{P}}(T) \subseteq \mathrm{M}_{\mathbb{P}}(\varphi)$, a uvědomíme-li si následující vztahy:

- $\mathrm{M}(\mathrm{Csq}(T)) = \mathrm{M}(T)$,
- je-li $T \subseteq T'$ potom $\mathrm{M}(T) \supseteq \mathrm{M}(T')$,[20]
- $\psi \to \varphi$ je tautologie, právě když platí $\mathrm{M}(\psi) \subseteq \mathrm{M}(\varphi)$,
- $\mathrm{M}(\varphi_1 \wedge \cdots \wedge \varphi_n) = \mathrm{M}(\varphi_1, \ldots, \varphi_n)$.

□

Cvičení 2.6. Dokažte podrobně Tvrzení 2.4.4.

[20] Čím více vlastností předepíšeme, tím méně objektů je bude všechny splňovat.

2.4.2 Extenze teorií

Neformálně řečeno, rozšířením, neboli *extenzí* teorie T myslíme jakoukoliv teorii T', která splňuje vše, co platí v teorii T (a něco navíc, nejde-li o triviální případ). Modeluje-li T nějaký systém, lze ji rozšířit dvěma způsoby: přidáním dodatečných požadavků o systému (tomu budeme říkat *jednoduchá extenze*) nebo i rozšířením systému o nějaké nové části. Pokud ve druhém případě nemáme dodatečné požadavky na původní část systému, tedy platí-li o původní části totéž, co předtím, říkáme, že je extenze *konzervativní*.

Příklad 2.4.5. Vraťme se k úvodnímu příkladu o barvení grafů, Příklad 1.1.2. Teorie T_3 (úplná obarvení grafu zachovávající hranovou podmínku) je jednoduchou extenzí teorie T_1 (částečná obarvení množiny vrcholů bez ohledu na hrany). Teorie T_3' z Sekce 1.2.1 (přidání nového vrcholu do grafu) je konzervativní, ale ne jednoduchou extenzí T_3. A jde o extenzi T_1, která není ani jednoduchá ani konzervativní.

Uveďme nyní konečně formální definice:

Definice 2.4.6 (Extenze teorie)**.** Mějme teorii T v jazyce $\mathbb{P}$.

- *Extenze* teorie T je libovolná teorie T' v jazyce $\mathbb{P}' \supseteq \mathbb{P}$ splňující $\mathrm{Csq}_{\mathbb{P}}(T) \subseteq \mathrm{Csq}_{\mathbb{P}'}(T')$,
- je to *jednoduchá extenze*, pokud $\mathbb{P}' = \mathbb{P}$,
- je to *konzervativní extenze*, pokud $\mathrm{Csq}_{\mathbb{P}}(T) = \mathrm{Csq}_{\mathbb{P}}(T') = \mathrm{Csq}_{\mathbb{P}'}(T') \cap \mathrm{VF}_{\mathbb{P}}$.

Extenze tedy znamená, že splňuje všechny důsledky původní teorie. Extenze je jednoduchá, pokud do jazyka nepřidáváme žádné nové výrokové proměnné, a konzervativní, pokud neměníme platnost tvrzení vyjádřitelných v původním jazyce, každý nový důsledek tedy musí obsahovat nějakou nově přidanou výrokovou proměnnou.

Co tyto pojmy znamenají *sémanticky*, v řeči modelů? Zformulujme nejprve obecné pozorování, které ihned poté ilustrujeme na příkladě:

Pozorování 2.4.7. *Je-li T teorie v jazyce $\mathbb{P}$ a T' teorie v jazyce $\mathbb{P}'$ obsahujícím jazyk $\mathbb{P}$. Potom platí:*

- *T' je jednoduchou extenzí T, právě když $\mathbb{P}' = \mathbb{P}$ a $\mathrm{M}_{\mathbb{P}}(T') \subseteq \mathrm{M}_{\mathbb{P}}(T)$,*
- *T' je extenzí T, právě když $\mathrm{M}_{\mathbb{P}'}(T') \subseteq \mathrm{M}_{\mathbb{P}'}(T)$. Uvažujeme tedy modely teorie T nad rozšířeným jazykem $\mathbb{P}'$.*[21] *Jinými slovy,* restrikce[22] *libovolného modelu $v \in \mathrm{M}_{\mathbb{P}'}(T')$ na původní jazyk $\mathbb{P}$ musí být modelem T, mohli bychom psát $v\restriction_{\mathbb{P}} \in \mathrm{M}_{\mathbb{P}}(T)$ nebo:*

$$\{v\restriction_{\mathbb{P}} \mid v \in \mathrm{M}_{\mathbb{P}'}(T')\} \subseteq \mathrm{M}_{\mathbb{P}}(T)$$

- *T' je konzervativní extenzí T, pokud je extenzí a navíc platí, že každý model T (v jazyce $\mathbb{P}$) lze nějak* expandovat *(rozšířit)*[23] *na model T' (v jazyce $\mathbb{P}'$), neboli* každý *model T (v jazyce $\mathbb{P}$) získáme restrikcí* nějakého *modelu T' na jazyk $\mathbb{P}$. Mohli bychom psát:*

$$\{v\restriction_{\mathbb{P}} \mid v \in \mathrm{M}_{\mathbb{P}'}(T')\} = \mathrm{M}_{\mathbb{P}}(T)$$

[21] Pozor, nemůžeme psát $\mathrm{M}_{\mathbb{P}}(T')$, protože modely T' musí být ohodnoceními většího jazyka $\mathbb{P}'$, hodnoty jen pro proměnné z $\mathbb{P}$ nestačí k určení pravdivostní hodnoty. A nelze psát ani $\mathrm{M}_{\mathbb{P}'}(T') \subseteq \mathrm{M}_{\mathbb{P}}(T)$, jde o množiny vektorů jiné dimenze.

[22] *Restrikce* znamená zapomenutí hodnot pro nové výrokové proměnné, resp. smazání příslušných souřadnic při reprezentaci modelu vektorem.

[23] Přidáním hodnot pro nové výrokové proměnné, resp. přidáním odpovídajících souřadnic ve vektorové reprezentaci.

- *T' je extenzí T a zároveň T je extenzí T', právě když $\mathbb{P}' = \mathbb{P}$ a $\mathrm{M}_{\mathbb{P}}(T') = \mathrm{M}_{\mathbb{P}}(T)$, neboli $T' \sim T$.*

- *Kompletní jednoduché extenze T jednoznačně až na ekvivalenci odpovídají modelům T.*

Příklad 2.4.8. Mějme teorii $T = \{p \to q\}$ v jazyce $\mathbb{P} = \{p, q\}$. Teorie $T_1 = \{p \wedge q\}$ v jazyce $\mathbb{P}$ je jednoduchou extenzí T, máme $\mathrm{M}_{\mathbb{P}}(T_1) = \{(1,1)\} \subseteq \{(0,0),(0,1),(1,1)\} = \mathrm{M}_{\mathbb{P}}(T)$. Je to kompletní teorie, další kompletní jednoduché extenze teorie T jsou např. $T_2 = \{\neg p, q\}$ a $T_3 = \{\neg p, \neg q\}$. Každá kompletní jednoduchá extenze teorie T je ekvivalentní s T_1, T_2, nebo T_3.

Uvažme nyní teorii $T' = \{p \leftrightarrow (q \wedge r)\}$ v jazyce $\mathbb{P}' = \{p, q, r\}$. Je extenzí T, neboť $\mathbb{P} = \{p, q\} \subseteq \{p, q, r\} = \mathbb{P}'$ a platí:

$$\begin{aligned}\mathrm{M}_{\mathbb{P}'}(T') &= \{(0,0,0),(0,0,1),(0,1,0),(1,1,1)\} \\ &\subseteq \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,1,0),(1,1,1)\} = \mathrm{M}_{\mathbb{P}'}(T)\end{aligned}$$

Jinými slovy, zúžením modelů T' na jazyk $\mathbb{P}$ dostáváme $\{(0,0),(0,1),(1,1)\}$ což je podmnožina $\mathrm{M}_{\mathbb{P}}(T)$.

Protože platí dokonce $\{(0,0),(0,1),(1,1)\} = \mathrm{M}_{\mathbb{P}}(T)$, jinými slovy, každý model $v \in \mathrm{M}_{\mathbb{P}}(T)$ lze rozšířit na model $v' \in \mathrm{M}_{\mathbb{P}'}(T')$ (např. $(0,1)$ lze rozšířit dodefinováním $v'(r) = 0$ na model $(0,1,0)$), je T' dokonce konzervativní extenzí T. To znamená, že každý výrok v jazyce $\mathbb{P}$ platí v T, právě když platí v T'. Ale výrok $p \to r$ (který je v jazyce $\mathbb{P}'$, ale ne v jazyce $\mathbb{P}$) je novým důsledkem: platí v T' ale ne v T (viz model $(1,1,0)$).

Teorie $T'' = \{\neg p \vee q, \neg q \vee r, \neg r \vee p\}$ v jazyce $\mathbb{P}'$ je extenzí T, ale ne konzervativní extenzí, neboť v ní platí $p \leftrightarrow q$, což neplatí v T. Nebo také proto, že model $(0,1)$ teorie T nelze rozšířit na model teorie T'': $(0,1,0)$ ani $(0,1,1)$ nesplňují axiomy T''.

Teorie T je (jednoduchou) extenzí teorie $\{\neg p \vee q\}$ v jazyce $\mathbb{P}$ a naopak, $T \sim \{\neg p \vee q\}$. Je také, jako každá teorie, jednoduchou konzervativní extenzí sebe sama.

Cvičení 2.7. Ukažte (podrobně), že má-li teorie T kompletní konzervativní extenzi, potom je sama nutně kompletní.

2.5 Algebra výroků

V logice nás většinou[24] zajímají výroky (resp. teorie) *až na ekvivalenci.*[25] Na otázku 'Kolik existuje různých výroků v jazyce $\mathbb{P} = \{p, q, r\}$?' je správná odpověď 'Nekonečně mnoho.' Nejspíše nás ale zajímaly výroky *až na ekvivalenci* (neboli *navzájem neekvivalentní*). Těch je tolik, kolik existuje různých podmnožin modelů jazyka, tedy $2^{|\mathrm{M}_{\mathbb{P}}|} = 2^8 = 256$. Skutečně, mají-li dva výroky stejnou množinu modelů, jsou z definice ekvivalentní. A pro každou množinu modelů můžeme najít odpovídající výrok, např. v DNF (viz 2.3.4). Zkusme trochu složitější úvahu:

Příklad 2.5.1. Mějme teorii T v jazyce $\mathbb{P} = \{p, q, r\}$ mající právě pět modelů. Kolik existuje (až na ekvivalenci) výroků nad $\mathbb{P}$, které jsou nezávislé v teorii T? Označme $|\mathbb{P}| = n = 3$ a $|\mathrm{M}_{\mathbb{P}}(T)| = k = 5$.

Počítáme množiny $M = \mathrm{M}_{\mathbb{P}}(\varphi)$ a požadujeme, aby $\emptyset \neq M \cap \mathrm{M}_{\mathbb{P}}(T) \neq \mathrm{M}_{\mathbb{P}}(T)$. Máme tedy celkem $2^k - 2 = 30$ možností, jak může vypadat množina $M \cap \mathrm{M}_{\mathbb{P}}(T)$. A pro každý model

[24] Pokud např. neprovádíme konkrétní algoritmus založený na syntaktických úpravách, třeba převod do CNF.

[25] Můžeme je chápat jako jakési abstraktní 'vlastnosti' modelů bez ohledu na jejich konkrétní vyjádření.

jazyka, který není modelem T (těch je $2^n - k = 3$) můžeme zvolit libovolně, zda bude či nebude v M. Celkově tedy dostáváme $(2^k - 2) \cdot 2^{2^n - k} = 30 \cdot 2^{8-5} = 240$ možných množin M, tolik je tedy výroků nezávislých v T, až na ekvivalenci.

Podívejme se na věc abstraktněji. Formálně, uvažujeme množinu ekvivalenčních tříd $\sim$ na množině všech výroků $\mathrm{VF}_{\mathbb{P}}$, kterou označíme ${}^{\mathrm{VF}_{\mathbb{P}}}/{\sim}$. Prvky této množiny jsou množiny ekvivalentních výroků, např. $[p{\to}q]_\sim = \{p{\to}q, \neg p{\vee}q, \neg(p{\wedge}\neg q), \neg p{\vee}q{\vee}q, \dots\}$. A máme zobrazení $h : {}^{\mathrm{VF}_{\mathbb{P}}}/{\sim} \to \mathcal{P}(\mathrm{M}_{\mathbb{P}})$ (kde $\mathcal{P}(X)$ je množina všech podmnožin X) definované předpisem:

$$h([\varphi]_\sim) = \mathrm{M}(\varphi)$$

tj. třídě ekvivalentních výroků přiřadíme množinu modelů libovolného z nich. Je snadné ověřit, že toto zobrazení je korektně definované (nezáleží na tom, jaký výrok z třídy ekvivalence jsme si vybrali) a prosté, a že je-li jazyk $\mathbb{P}$ konečný, je h dokonce bijekce. (Ověřte!)

Na množině ${}^{\mathrm{VF}_{\mathbb{P}}}/{\sim}$ můžeme zavést operace $\neg, \wedge, \vee$ pomocí předpisu

$$\begin{aligned} \neg[\varphi]_\sim &= [\neg\varphi]_\sim \\ [\varphi]_\sim \wedge [\psi]_\sim &= [\varphi \wedge \psi]_\sim \\ [\varphi]_\sim \vee [\psi]_\sim &= [\varphi \vee \psi]_\sim \end{aligned}$$

tedy vybereme reprezentanta resp. reprezentanty, a provedeme operaci s nimi, např. 'konjunkce' tříd $[p \to q]_\sim$ a $[q \vee \neg r]_\sim$ je:

$$[p \to q]_\sim \wedge [q \vee \neg r]_\sim = [(p \to q) \wedge (q \vee \neg r)]_\sim$$

Přidáme-li také *konstanty* $\bot = [\bot]_\sim$ a $\top = [\top]_\sim$, dostáváme *(matematickou) strukturu*[26]

$$\mathbf{AV}_{\mathbb{P}} = \langle {}^{\mathrm{VF}_{\mathbb{P}}}/{\sim}; \neg, \wedge, \vee, \bot, \top \rangle$$

které říkáme *algebra výroků* jazyka $\mathbb{P}$. Je to příklad tzv. *Booleovy algebry*. To znamená, že její operace se 'chovají' jako operace $^-, \cap, \cup$ na množině všech podmnožin $\mathcal{P}(X)$ nějaké neprázdné množiny X, a konstanty odpovídají $\emptyset$, X (takové Booleově algebře říkáme *potenční algebra*).[27]

Zobrazení $h : {}^{\mathrm{VF}_{\mathbb{P}}}/{\sim} \to \mathcal{P}(\mathrm{M}_{\mathbb{P}})$ je tedy prosté zobrazení z algebry výroků $\mathbf{AV}_{\mathbb{P}}$ do potenční algebry

$$\mathcal{P}(\mathrm{M}_{\mathbb{P}}) = \langle \mathcal{P}(\mathrm{M}_{\mathbb{P}}); {}^-, \cap, \cup, \emptyset, \mathrm{M}_{\mathbb{P}} \rangle$$

a je-li jazyk konečný, je to bijekce. Toto zobrazení 'zachovává' operace a konstanty, tj. platí $h(\bot) = \emptyset$, $h(\top) = \mathrm{M}_{\mathbb{P}}$, a

$$\begin{aligned} h(\neg[\varphi]_\sim) &= \overline{h([\varphi]_\sim)} = \overline{\mathrm{M}(\varphi)} = \mathrm{M}_{\mathbb{P}} \setminus \mathrm{M}(\varphi) \\ h([\varphi]_\sim \wedge [\psi]_\sim) &= h([\varphi]_\sim) \cap h([\psi]_\sim) = \mathrm{M}(\varphi) \cap \mathrm{M}(\psi) \\ h([\varphi]_\sim \vee [\psi]_\sim) &= h([\varphi]_\sim) \cup h([\psi]_\sim) = \mathrm{M}(\varphi) \cup \mathrm{M}(\psi) \end{aligned}$$

Takovému zobrazení říkáme *homomorfismus* Booleových algeber, a je-li to bijekce, jde o *izomorfismus*.

[26] Struktura je neprázdná množina spolu s relacemi, operacemi, a konstantami. Například (orientovaný) graf, grupa, těleso, vektorový prostor. Struktury budou hrát důležitou roli v predikátové logice.

[27] Tj. splňují určité algebraické zákony, například distributivitu $\wedge$ vůči $\vee$. Booleovy algebry definujeme formálně později, uveďme ale ještě jeden důležitý příklad: množina všech n-bitových vektorů s operacemi $\sim$, &, | (po složkách) a s konstantami $(0, 0, \dots, 0)$ a $(1, 1, \dots, 1)$.

Poznámka 2.5.2. Tyto vztahy můžeme také využít při hledání modelů: například pro výrok $\varphi \to (\neg\psi \wedge \chi)$ platí (s využitím toho, že $\mathrm{M}(\varphi \to \varphi') = M(\neg\varphi \vee \varphi')$):

$$\mathrm{M}(\varphi \to (\neg\psi \wedge \chi)) = \overline{\mathrm{M}(\varphi)} \cup (\overline{\mathrm{M}(\psi)} \cap \mathrm{M}(\chi))$$

Všechny předchozí úvahy můžeme také relativizovat vzhledem k dané teorii T v jazyce $\mathbb{P}$, a to tak, že ekvivalenci $\sim$ nahradíme T-ekvivalencí $\sim_T$ a množinu modelů jazyka $\mathrm{M}_\mathbb{P}$ nahradíme množinou modelů teorie $\mathrm{M}_\mathbb{P}(T)$. Dostáváme:

$$\begin{aligned} h(\bot) &= \emptyset, \\ h(\top) &= \mathrm{M}(T) \\ h(\neg[\varphi]_{\sim_T}) &= \mathrm{M}(T) \setminus \mathrm{M}(T, \varphi) \\ h([\varphi]_{\sim_T} \wedge [\psi]_{\sim_T}) &= \mathrm{M}(T, \varphi) \cap \mathrm{M}(T, \psi) \\ h([\varphi]_{\sim_T} \vee [\psi]_{\sim_T}) &= \mathrm{M}(T, \varphi) \cup \mathrm{M}(T, \psi) \end{aligned}$$

Výslednou *algebru výroků vzhledem k teorii* T označíme $\mathbf{AV}_\mathbb{P}(T)$. Algebra výroků jazyka je tedy totéž co algebra výroků vzhledem k prázdné teorii. Z technických důvodů potřebujeme, aby $\mathrm{M}(T)$ byla neprázdná, tj. T musí být bezesporná. Shrňme naše úvahy:

Důsledek 2.5.3. *Je-li T bezesporná teorie nad* konečným *jazykem $\mathbb{P}$, potom je algebra výroků $\mathbf{AV}_\mathbb{P}(T)$ izomorfní potenční algebře $\mathcal{P}(\mathrm{M}_\mathbb{P}(\mathbf{T}))$ prostřednictvím zobrazení $h([\varphi]_{\sim_T}) = M(T, \varphi)$.*

Víme tedy, že negace, konjunkce, a disjunkce odpovídají doplňku, průniku a sjednocení množin modelů, a že chceme-li najít počet výroků až na ekvivalenci resp. T-ekvivalenci, stačí určit počet příslušných množin modelů. Shrňme si několik takových výpočtů ve formě tvrzení, jeho důkaz necháme jako cvičení.

Tvrzení 2.5.4. *Mějme n-prvkový jazyk $\mathbb{P}$ a bezespornou teorii T mající právě k modelů. Potom v jazyce $\mathbb{P}$ existuje* až na ekvivalenci*:*

- 2^{2^n} *výroků (resp. teorií),*
- 2^{2^n-k} *výroků pravdivých (resp. lživých) v T,*
- $2^{2^n} - 2 \cdot 2^{2^n-k}$ *výroků nezávislých v T,*
- 2^k *jednoduchých extenzí teorie T (z toho 1 sporná),*
- k *kompletních jednoduchých extenzí T.*

Dále až na T-ekvivalenci *existuje:*

- 2^k *výroků,*
- 1 *výrok pravdivý v T,* 1 *lživý v T,*
- $2^k - 2$ *výroků nezávislých v T.*

Cvičení 2.8. Zvolte vhodnou teorii T a ukažte na jejím příkladě, že platí Tvrzení 2.5.4.

Cvičení 2.9. Dokažte podrobně Tvrzení 2.5.4. (Nakreslete si Vennův diagram.)

Cvičení 2.10. Dokažte podrobně, že zobrazení h z Důsledku 2.5.3 je korektně definované, prosté, a je-li jazyk konečný, potom i na.

Kapitola 3

Problém splnitelnosti

Problém splnitelnosti výrokových formulí, známý také jako *problém SAT*[1] je následující výpočetní problém: Vstupem je výrok φ v CNF (v nějakém rozumném kódování[2]), a úkolem je rozhodnout, zda je φ *splnitelný*. [3]

Jak jsme si ukázali v předchozí kapitole, můžeme každý výrok, nebo i každou výrokovou teorii v konečném jazyce, převést na CNF formuli. Problém SAT je tedy v jistém smyslu univerzální; odpovídá na otázku, zda existuje model.

Známá Cook-Levinova věta říká, že problém SAT je *NP-úplný*, tedy je v třídě NP (pokud nám orákulum prozradí správné ohodnocení proměnných, můžeme snadno ověřit, že všechny klauzule jsou splněny) a každý problém z třídy NP na něj lze převést v polynomiálním čase (konkrétně, výpočet Turingova stroje lze popsat pomocí výroku v CNF).[4]

Praktické SAT solvery si ale umí poradit s instancemi obsahujícími mnoho (až desítky milionů) výrokových proměnných a klauzulí. V této kapitole si nejprve ukážeme praktickou aplikaci SAT solveru na problém 'ze života', potom dva fragmenty problému SAT, tzv. *2-SAT* a *Horn-SAT*, pro které existují polynomiální algoritmy, a na závěr si ukážeme také algoritmus DPLL, který je základem (téměř) všech SAT solverů. (Později, v Kapitole 4, uvidíme také souvislost s *rezoluční metodou*.)

3.1 SAT solvery

První řešiče SAT byly vyvinuty v 60. letech 20. století. Jejich základem je téměř vždy algoritmus DPLL (Davis–Putnam–Logemann–Loveland), který představíme v Sekci 3.4, respektive některé z jeho vylepšení. Po roce 2000 dochází k poněkud překvapivému, dramatickému vývoji technologií pro řešiče SAT a tím i k rapidnímu růstu jejich užitečnosti v různých oblastech aplikované informatiky.

Moderní SAT solvery používají celou řadu technologií pro efektivní řešení typických instancí pocházejících z různých aplikačních domén, strategií a heuristik pro exploraci prostoru řešení (například i za použití strojového učení a neuronových sítí), a dalších vylepšení. Tyto moderní nástroje mají typicky několik desítek tisíc řádků kódu. Dostupnost efektivních SAT

[1] Z anglického 'Boolean satisfiability problem'.

[2] Např. formát DIMACS-CNF.

[3] Pozor, v některé literatuře se jako SAT označuje splnitelnost *libovolného* výroku, a na CNF je potom omezen až problém k-SAT (viz níže).

[4] Viz předmět NTIN090 Základy složitosti a vyčíslitelnosti.

solverů významně ovlivnila vývoj například v oblasti softwarové verifikace, analýzy programů, optimalizace, nebo umělé inteligence. Nejlepší SAT solvery spolu pravidelně soutěží v rámci SAT competition.

Pro vyzkoušení SAT solvingu nám poslouží řešič `Glucose`. Ten přijímá vstup v jednoduchém formátu DIMACS CNF. Ukažme si postup použití na následující hříčce zvané *boardomino*:

Příklad 3.1.1 (Boardomino). Lze pokrýt šachovnici s chybějícími dvěma protilehlými rohy perfektně pokrýt kostkami domina?

Jak tento problém formalizovat? Zvolme výrokové proměnné $h_{i,j}, v_{i,j}$ $(1 \leq i, j \leq n)$, kde $h_{i,j}$ znamená "na pozici (i, j) leží levá polovina horizontálně orientované kostky" a podobně $v_{i,j}$ pro horní polovinu vertikální kostky. Zde $n = 8$, ale můžeme vyzkoušet i pro jiné (sudé) rozměry šachovnice. Nyní axiomatizujeme všechny požadované vlastnosti:

- levý horní a pravý dolní roh chybí: $\neg h_{11}, \neg v_{11}, \neg h_{n,n-1}, \neg v_{n-1,n}$
- kostky nevyčnívají z šachovnice (vpravo ani dolů): $\neg h_{i,n}, \neg v_{n,i}$ pro $1 \leq i \leq n$
- každé políčko je pokryto alespoň jednou kostkou (první řádek a sloupec zvlášť):

$$\begin{gathered} h_{i,j-1} \vee h_{i,j} \vee v_{i-1,j} \vee v_{i,j} \text{ pro } 1 < i, j \leq n \\ h_{1,j-1} \vee h_{1,j} \vee v_{1,j} \text{ pro } 1 < j \leq n \\ h_{i,1} \vee v_{i-1,1} \vee v_{i,1} \text{ pro } 1 < i \leq n \end{gathered}$$

- každé políčko je pokryto nejvýše jednou kostkou (první řádek a sloupec zvlášť):

$$\begin{gathered} (\neg h_{i,j-1} \vee \neg h_{i,j}) \wedge (\neg h_{i,j-1} \vee \neg v_{i-1,j}) \wedge (\neg h_{i,j-1} \vee \neg v_{i,j}) \wedge \\ (\neg h_{i,j} \vee \neg v_{i-1,j}) \wedge (\neg h_{i,j} \vee \neg v_{i,j}) \wedge (\neg v_{i-1,j} \vee \neg v_{i,j}) \text{ pro } 1 < i, j \leq n \\ (\neg h_{1,j-1} \vee \neg h_{1,j}) \wedge (\neg h_{1,j-1} \vee \neg v_{1,j}) \wedge (\neg h_{1,j} \vee \neg v_{1,j}) \text{ pro } 1 < j \leq n \\ (\neg h_{i,1} \vee \neg v_{i-1,1}) \wedge (\neg h_{i,1} \vee \neg v_{i,1}) \wedge (\neg v_{i-1,1} \vee \neg v_{i,1}) \text{ pro } 1 < i \leq n \end{gathered}$$

Výsledná teorie už je v CNF, snadno ji můžeme zapsat ve formátu DIMACS CNF, a vyřešit pomocí solveru Glucose. V praxi bychom mohli tento převod naprogramovat, nebo využít jednoho z mnoha vysokoúrovňových jazyků z oblasti *constraint programming* umožňujících překlad do SATu.

Uvidíme, že takové instance problému SAT budou pro řešiče těžké a už pro poměrně malé rozměry šachovnice se řešení nedočkáme. Jako matematici snadno nahlédneme, že řešení neexistuje: Každá kostka domina pokrývá jedno bílé a jedno černé políčko, ale odebrali jsme dvě bílá, nutně tedy zbudou dvě černá. Tento pohled ale není v zakódování do CNF dostupný. Lze najít částečná ohodnocení téměř všech proměnných, aniž bychom nějakou podmínku porušili. Solver tedy bude muset prohledat téměř celý prostor řešení, než dokáže nesplnitelnost.[5] Klíčovým náhledem do SAT solvingu je fakt, že takové těžké instance se v praxi téměř nikdy nevyskytují.

[5] Podobné vlastnosti má také zakódování *holubníkového principu* do SATu.

3.2 2-SAT a implikační graf

Výrok φ je v k-CNF, pokud je v CNF a každá klauzule má nejvýše k literálů. Problému k-SAT se ptá, zda je daný k-CNF výrok splnitelná. Pro $k \geq 3$ je k-SAT nadále NP-úplný, každou CNF formuli lze zakódovat do 3-CNF výroku:

Cvičení 3.1. Ukažte, že pro každý výrok φ v CNF existuje *ekvisplnitelný* výrok v φ' 3-CNF (tj. φ je splnitelný, právě když φ' je splnitelný), který lze zkonstruovat v lineárním čase.

Pro problém 2-SAT ale existuje polynomiální (dokonce lineární) algoritmus, který si nyní představíme. Algoritmus využívá tzv. *implikačního grafu*. Ukážeme si postup na příkladě:

Příklad 3.2.1. Mějme následující 2-CNF výrok φ:

$$(\neg p_1 \vee p_2) \wedge (\neg p_2 \vee \neg p_3) \wedge (p_1 \vee p_3) \wedge (p_3 \vee \neg p_4) \wedge (\neg p_1 \vee p_5) \wedge (p_2 \vee p_5) \wedge p_1 \wedge \neg p_4$$

Implikační graf

Implikační graf 2-CNF výroku φ je založený na myšlence, že 2-klauzuli $\ell_1 \vee \ell_2$ (kde ℓ_1, ℓ_2 jsou literály) lze chápat jako dvojici implikací: $\overline{\ell_1} \to \ell_2$ a $\overline{\ell_2} \to \ell_1$.[6] Například, z klauzule $\neg p_1 \vee p_2$ vzniknou implikace $p_1 \to p_2$ a také , $\neg p_2 \to \neg p_1$. Tedy pokud p_1 platí v nějakém modelu, musí platit i p_2, a pokud p_2 neplatí, nesmí platit ani p_1. Jednotkovou klauzuli ℓ můžeme také vyjádřit pomocí implikace jako $\overline{\ell} \to \ell$, např. z p_1 dostáváme $\neg p_1 \to p_1$.

Implikační graf $\mathcal{G}_\varphi$ je tedy orientovaný graf, jehož vrcholy jsou všechny literály (proměnné z $\mathrm{Var}(\varphi)$ a jejich negace) a hrany jsou dané implikacemi popsanými výše:

- $V(\mathcal{G}_\varphi) = \{p, \neg p \mid p \in \mathrm{Var}(\varphi)\}$,
- $E(\mathcal{G}_\varphi) = \{(\overline{\ell_1}, \ell_2), (\overline{\ell_2}, \ell_1) \mid \ell_1 \vee \ell_2 \text{ je klauzule } \varphi\} \cup \{(\overline{\ell}, \ell) \mid \ell \text{ je jednotková klauzule } \varphi\}$

V našem příkladě máme množinu vrcholů

$$V(\mathcal{G}_\varphi) = \{p_1, p_2, p_3, p_4, p_5, \neg p_1, \neg p_2, \neg p_3, \neg p_4, \neg p_5\}$$

a hrany jsou:

$$E(\mathcal{G}_\varphi) = \{(p_1, p_2), (\neg p_2, \neg p_1), (p_2, \neg p_3), (p_3, \neg p_2), (\neg p_1, p_3), (\neg p_3, p_1), (\neg p_3, \neg p_4), \\ (p_4, p_3), (p_1, p_5), (\neg p_5, \neg p_1), (\neg p_2, p_5), (\neg p_5, p_2), (\neg p_1, p_1), (p_4, \neg p_4)\}$$

Výsledný graf je znázorněný na Obrázku 3.1.

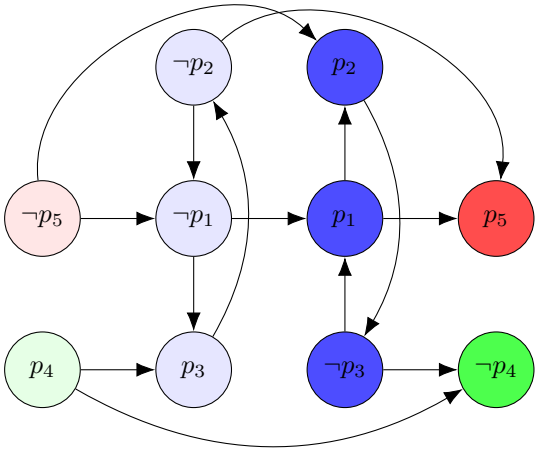
3.2.1 Silně souvislé komponenty

Nyní musíme najít komponenty silné souvislosti[7] tohoto grafu. V našem příkladě dostáváme následující komponenty: $C_1 = \{p_4\}$, $C_2 = \{\neg p_5\}$, $C_3 = \{\neg p_1, \neg p_2, p_3\}$, $\overline{C_3} = \{p_1, p_2, \neg p_3\}$, $\overline{C_2} = \{p_5\}$, $\overline{C_1} = \{\neg p_4\}$.

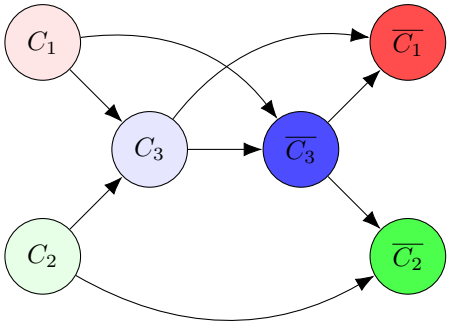
Všechny literály v jedné komponentě musí být ohodnoceny stejně. Pokud bychom tedy našli dvojici opačných literálů v jedné komponentě, znamená to, že výrok je nesplnitelný. V opačném případě vždy můžeme najít splňující ohodnocení, jak si dokážeme v Tvrzení 3.2.2.

[6] V předchozí kapitole jsme vyjadřovali $p_1 \to p_2$ jako $\neg p_1 \vee p_2$, zde provádíme opačný postup.

[7] *Silná souvislost* znamená, že existuje orientovaná cesta z u do v i z v do u, neboli každé dva vrcholy v jedné komponentě leží v orientovaném cyklu. A naopak, každý orientovaný cyklus leží uvnitř nějaké komponenty.



Obrázek 3.1: Implikační graf $\mathcal{G}_\varphi$. Komponenty silné souvislosti jsou odlišeny barevně.



Obrázek 3.2: Implikační graf $\mathcal{G}_\varphi$. Graf silně souvislých komponent $\mathcal{G}^*_\varphi$.

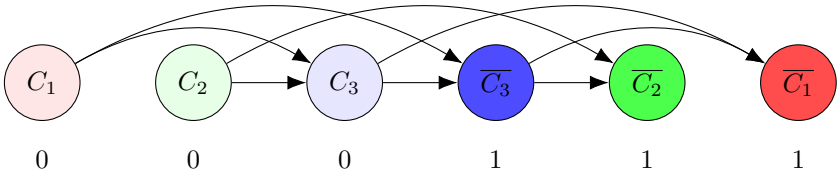
Potřebujeme zajistit, aby z žádné komponenty ohodnocené 1 nevedla hrana do komponenty ohodnocené 0. Provedeme-li kontrakci komponent (a odstraníme-li smyčky), výsledný graf $\mathcal{G}^*_\varphi$ je acyklický (každý cyklus byl uvnitř nějaké komponenty), viz Obrázek 3.2. To znamená, že ho můžeme nakreslit v *topologickém uspořádání* (tj. uspořádání na přímce, kde hrany vedou jen doprava), viz Obrázek 3.3 níže.

Při hledání splňujícího ohodnocení (pokud nám nestačí informace, že výrok je splnitelný) potom postupujeme tak, že vezmeme nejlevější dosud neohodnocenou komponentu, ohodnotíme ji 0, opačnou komponentu ohodnotíme 1, a postup opakujeme dokud zbývá nějaká neohodnocená komponenta. Například, topologické uspořádání na Obrázku 3.3 odpovídá modelu $v = (1, 1, 0, 0, 1)$.

Na závěr shrneme naše úvahy do následujícího tvrzení:

Tvrzení 3.2.2. *Výrok φ je splnitelný, právě když žádná silně souvislá komponenta v $\mathcal{G}_\varphi$ neobsahuje dvojici opačných literálů $\ell, \overline{\ell}$.*

Důkaz. Každý model, neboli splňující ohodnocení, musí ohodnotit všechny literály ze stejné komponenty stejnou hodnotou. (V opačném případě by nutně existovala implikace $\ell_1 \to \ell_2$, kde ℓ_1 v modelu platí ale ℓ_2 neplatí.) V jedné komponentě tedy nemohou být opačné literály.



Obrázek 3.3: Implikační graf $\mathcal{G}_\varphi$. Topologické uspořádání grafu $\mathcal{G}_\varphi^*$ a splňující ohodnocení komponent.

Naopak předpokládejme, že žádná komponenta neobsahuje dvojici opačných literálů, a ukažme, že potom existuje model. Označme $\mathcal{G}_\varphi^*$ graf vzniklý z $\mathcal{G}_\varphi$ kontrakcí silně souvislých komponent (a odstraněním smyček). Tento graf je acyklický, zvolme nějaké topologické uspořádání. Model zkonstruujeme tak, že zvolíme první dosud neohodnocenou komponentu v našem topologickém uspořádání, všechny literály v ní obsažené ohodnotíme 0, a opačné literály ohodnotíme 1. Takto pokračujeme dokud nejsou všechny komponenty ohodnoceny.

Proč v takto získaném modelu platí výrok φ? Kdyby ne, neplatila by některá z klauzulí. Jednotková klauzule ℓ musí platit, neboť v grafu $\mathcal{G}_\varphi$ máme hranu $\overline{\ell} \to \ell$. Stejná hrana je i v grafu komponent, tedy $\overline{\ell}$ předchází v topologickém uspořádání komponentu obsahující ℓ. Při konstrukci modelu jsme museli ohodnotit $\overline{\ell}$ dříve než ℓ, tedy $\overline{\ell} = 0$ a $\ell = 1$. Podobně, 2-klauzule $\ell_1 \vee \ell_2$ také musí platit: máme hrany $\overline{\ell_1} \to \ell_2$ a $\overline{\ell_2} \to \ell_1$. Pokud jsme ℓ_1 ohodnotili dříve než ℓ_2, museli jsme kvůli hraně $\overline{\ell_1} \to \ell_2$ ohodnotit $\overline{\ell_1} = 0$, tedy ℓ_1 platí. Podobně pokud jsme ohodnotili nejdříve ℓ_2, musí být $\overline{\ell_2} = 0$ a $\ell_2 = 1$. □

Důsledek 3.2.3. *Problém 2-SAT je řešitelný v lineárním čase. V lineárním čase můžeme také zkonstruovat model, pokud existuje.*

Důkaz. Komponenty silné souvislosti lze snadno nalézt v čase $\mathcal{O}(|V|+|E|)$, topologické uspořádání můžeme také zkonstruovat v čase $\mathcal{O}(|V|+|E|)$. □

Cvičení 3.2. Najděte nějaký nesplnitelný 2-CNF výrok, sestrojte jeho implikační graf, a přesvědčete se, že existuje dvojice opačných literálů ve stejné komponentě silné souvislosti.

Cvičení 3.3. Najděte všechna topologická uspořádání grafu $\mathcal{G}_\varphi^*$ z příkladu výše a jim odpovídající modely. Rozmyslete si, proč takto získáme právě všechny modely výroku φ.

Cvičení 3.4. Rozmyslete si, proč lze komponenty i topologické uspořádání nalézt v čase $\mathcal{O}(|V|+|E|)$.

3.3 Horn-SAT a jednotková propagace

Nyní si ukážeme další fragment SATu řešitelný v polynomiálním čase, tzv. *Horn-SAT* neboli problém splnitelnosti *hornovských výroků*. Výrok je v *hornovský (v Hornově tvaru)*[8], pokud je konjunkcí *hornovských klauzulí*, tj. klauzulí obsahujících *nejvýše jeden *pozitivní* literál*. Význam Hornovských klauzulí vyplývá z ekvivalentního vyjádření ve formě implikace:

$$\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n \vee q \;\sim\; (p_1 \wedge p_2 \wedge \cdots \wedge p_n) \to q$$

[8]Matematik Alfred Horn objevil význam tohoto tvaru logických formulí (a položil tak základ logickému programování) v roce 1951.

Hornovské výroky tedy dobře modelují systémy, kde splnění určitých podmínek zaručuje splnění jiné podmínky. Upozorněme, že jednotková klauzule ℓ je také hornovská. V kontextu logického programování se jí říká *fakt*, pokud je literál pozitivní, a *cíl* pokud je negativní.[9] Hornovské výroky s alespoň jedním pozitivním a alespoň jedním negativním literálem jsou *pravidla*.

Příklad 3.3.1. Příkladem výroku, který je v CNF, ale není hornovský, je třeba $(p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p_1 \vee p_3)$. Jako příklad, na kterém budeme ilustrovat algoritmus, nám poslouží následující hornovský výrok:

$$\varphi = (\neg p_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_2 \vee \neg p_3) \wedge (\neg p_5 \vee \neg p_4) \wedge p_4$$

Polynomiální algoritmus pro řešení problému Horn-SAT je založený na jednoduché myšlence *jednotkové propagace*: Pokud náš výrok obsahuje *jednotkovou* klauzuli, víme, jak musí být ohodnocena výroková proměnná obsažená v této klauzuli. A tuto znalost můžeme *propagovat*—využít k zjednodušení výroku.

Náš výrok φ obsahuje jednotkovou klauzuli p_4. Víme tedy, že v každém jeho modelu $v \in \mathrm{M}(\varphi)$ musí platit $v(p_4) = 1$. To ale znamená, že v libovolném modelu výroku φ

- každá klauzule obsahující pozitivní literál p_4 je splněna, můžeme ji tedy z výroku odstranit,
- negativní literál $\neg p_4$ nemůže být splněn, můžeme ho tedy odstranit ze všech klauzulí, které ho obsahují.

Tomu kroku se říká *jednotková propagace*. Výsledkem je následující zjednodušený výrok, který označíme φ^{p_4} (obecně φ^ℓ máme-li jednotkovou klauzuli ℓ):

$$\varphi^{p_4} = (\neg p_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_2 \vee \neg p_3) \wedge \neg p_5$$

Pozorování 3.3.2. *Všimněte si, že φ^ℓ už neobsahuje literál ℓ ani $\overline{\ell}$, a zřejmě platí, že modely φ jsou právě modely $\{\varphi^\ell, \ell\}$, neboli modely φ^ℓ v původním jazyce $\mathbb{P}$, ve kterých platí ℓ.*

Jednotkovou propagací jsme získali ve výroku φ^{p_4} novou jednotkovou klauzuli $\neg p_5$, můžeme tedy pokračovat nastavením $v(p_5) = 0$ a další jednotkovou propagací:

$$(\varphi^{p_4})^{\neg p_5} = (\neg p_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_2 \vee \neg p_3)$$

Výsledný výrok už neobsahuje jednotkovou klauzuli. To ale znamená, že každá klauzule obsahuje alespoň dva literály, a nejvýše jeden z nich může být pozitivní! (Zde potřebujeme hornovskost výroku.) Protože každá klauzule obsahuje negativní literál, stačí ohodnotit všechny zbývající proměnné 0, a výrok bude splněn: $v(p_1) = v(p_2) = v(p_3) = 0$. Dostáváme tedy model $v = (0, 0, 0, 1, 0)$.

Příklad 3.3.3. Co by se stalo, pokud by výrok nebyl splnitelný? Podívejme se na výrok

$$\psi = p \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$$

a provádějme jednotkovou propagaci jako v předchozím příkladě: máme $v(p) = 1$ a $\psi^p = q \wedge (\neg q \vee r) \wedge \neg r$, dále $v(q) = 1$ a $(\psi^p)^q = r \wedge \neg r$. Tento výrok je nesplnitelný, neboť obsahuje dvojici opačných jednotkových klauzulí. [10]

[9] Neboť dokazujeme sporem, více v pozdější kapitole o rezoluci a Prologu.

[10] Jinými slovy, v dalším kroku bychom provedli jednotkovou propagaci r, odstranili jednotkovou klauzuli r, a ze zbývající jednotkové klauzule $\neg r$ bychom odstranili literál $\neg r$, čímž by vznikla *prázdná klauzule*, která je nesplnitelná.

Shrňme si nyní algoritmus pro řešení problému Horn-SAT:

Algoritmus (Horn-SAT)**. vstup**: výrok φ v Hornově tvaru, **výstup**: model φ nebo informace, že φ není splnitelný

1. Pokud φ obsahuje dvojici opačných jednotkových klauzulí $\ell, \overline{\ell}$, není splnitelný.
2. Pokud φ neobsahuje žádnou jednotkovou klauzuli, je splnitelný, ohodnoť všechny zbývající proměnné 0.
3. Pokud φ obsahuje jednotkovou klauzuli ℓ, ohodnoť literál ℓ hodnotou 1, proveď jednotkovou propagaci, nahraď φ výrokem φ^ℓ, a vrať se na začátek.

Tvrzení 3.3.4. *Algoritmus je korektní.*

Důkaz. Korektnost plyne z Pozorování a z předchozí diskuze. □

Důsledek 3.3.5. *Horn-SAT lze řešit v lineárním čase.*

Důkaz. V každém kroku stačí projít výrok jednou, a jednotková propagace výrok vždy zkrátí. Z toho snadno plyne kvadratický horní odhad, ale při vhodné implementaci lze dosáhnout lineárního času vzhledem k délce φ. □

Cvičení 3.5. Navrhněte implementaci algoritmu pro Horn-SAT v lineárním čase.

Cvičení 3.6. Navrhněte modifikaci algoritmu pro Horn-SAT, která najde všechny modely.

3.4 DPLL algoritmus pro řešení problému SAT

Na závěr kapitoly o problému splnitelnosti si představíme zdaleka nejpoužívanější algoritmus pro řešení obecného problému SAT, algoritmus DPLL.[11] Ačkoliv v nejhorším případě má exponenciální složitost, v praxi funguje velmi efektivně.

Algoritmus používá jednotkovou propagaci spolu s následujícím pozorováním: Řekneme, že literál ℓ má *čistý výskyt* v φ, pokud se vyskytuje ve φ, ale opačný literál $\overline{\ell}$ se ve φ nevyskytuje. Máme-li literál s čistým výskytem, můžeme jeho hodnotu nastavit na 1, a splnit (a odstranit) tak všechny klauzule, které ho obsahují. Pokud výrok neumíme takto zjednodušit, rozvětvíme výpočet dosazením obou možných hodnot pro vybranou výrokovou proměnnou.

Algoritmus (DPLL)**. vstup**: výrok φ v CNF, **výstup**: model φ nebo informace, že φ není splnitelný

1. Dokud φ obsahuje jednotkovou klauzuli ℓ, ohodnoť literál ℓ hodnotou 1, proveď jednotkovou propagaci, a nahraď φ výrokem φ^ℓ.
2. Dokud existuje literál ℓ, který má ve φ čistý výskyt, ohodnoť ℓ hodnotou 1, a odstraň klauzule obsahující ℓ.
3. Pokud φ neobsahuje žádnou klauzuli, je splnitelný.
4. Pokud φ obsahuje prázdnou klauzuli, není splnitelný.

[11] Pojmenovaný po svých tvůrcích, Davis-Putnam-Logemann-Loveland, pochází z roku 1961.

5. Jinak zvol dosud neohodnocenou výrokovou proměnnou p, a zavolej algoritmus rekurzivně na $\varphi \wedge p$ a na $\varphi \wedge \neg p$.

Algoritmus běží v exponenciálním čase: počet větvení výpočtu nemůže být větší než počet proměnných. Lze ukázat, že v nejhorším případě je opravdu potřeba exponenciální čas. Korektnost algoritmu není těžké ověřit.

Tvrzení 3.4.1. *Algoritmus DPLL řeší problém SAT.*

Příklad 3.4.2. Ukážeme si běh algoritmu na následujícím příkladě:

$$(\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee \neg s) \wedge (p \vee \neg r \vee \neg s) \wedge (q \vee \neg r \vee s) \wedge (p \vee s) \wedge (p \vee \neg s) \wedge (q \vee s)$$

Výrok nemá žádnou jednotkovou klauzuli. Literál $\neg r$ má čistý výskyt, nastavíme $v(r) = 0$ a odstraníme klauzule obsahující $\neg r$:

$$(\neg p \vee \neg q \vee \neg s) \wedge (p \vee s) \wedge (p \vee \neg s) \wedge (q \vee s)$$

Žádný další literál nemá čistý výskyt. Spustíme proto rekurzivně algoritmus:

(p=1) Přidáme jednotkovou klauzuli p:

$$(\neg p \vee \neg q \vee \neg s) \wedge (p \vee s) \wedge (p \vee \neg s) \wedge (q \vee s) \wedge p$$

Nastavíme $v(p) = 1$ a provedeme jednotkovou propagaci: $(\neg q \vee \neg s) \wedge (q \vee s)$. Nyní rozvětvíme na proměnné q:

(q=1) $(\neg q \vee \neg s) \wedge (q \vee s) \wedge q$. Po nastavení $v(q) = 1$ a jednotkové propagaci dostáváme $\neg s$, po nastavení $v(s) = 0$ a jednotkové propagaci dostáváme výrok neobsahující žádnou klauzuli, je tedy splnitelný ohodnocením $(1, 1, 0, 0)$. Odpověď na problém splnitelnosti už máme, ostatní větve výpočtu nemusíme dokončovat. Pro ilustraci to ale provedeme.

(q=0) $(\neg q \vee \neg s) \wedge (q \vee s) \wedge \neg q$. Jednotkovou propagací s $v(q) = 0$ dostáváme s, po nastavení $v(s) = 1$ a jednotkové propagaci máme prázdnou množinu klauzulí. Dostáváme model $(1, 0, 0, 1)$.

(p=0) Přidáme jednotkovou klauzuli $\neg p$:

$$(\neg p \vee \neg q \vee \neg s) \wedge (p \vee s) \wedge (p \vee \neg s) \wedge (q \vee s) \wedge \neg p$$

Po provedení jednotkové propagace $\neg p$ máme $s \wedge \neg s \wedge (q \vee s)$. Po provedení jednotkové propagace s máme $\square \wedge q$, kde $\square$ je prázdná klauzule. Výrok je tedy nesplnitelný a v této větvi nedostaneme žádné modely.

Zjistili jsme, že původní výrok je splnitelný. Našli jsme 2 modely: $(1, 1, 0, 0)$ a $(1, 0, 0, 1)$. Mohou ale existovat i další modely, ohodnocení $v(r) = 0$ pro literál $\neg r$ s čistým výskytem nemusí být nutné pro splnění všech klauzulí; tento krok nezachovává množinu modelů, jen splnitelnost.

Co dále? Základní algoritmus DPLL, který systematicky prohledává prostor možných řešení, byl na konci 90. let 20. století různými způsoby doplněn a rozšířen. Zmiňme algoritmus zvaný *Conflict-driven clause learning (CDCL)*. Ten je založený na myšlence, že ze selhání větve prohledávacího stromu se můžeme *naučit* novou klauzuli, která tomuto konkrétnímu selhání ("konfliktu") zabraňuje. Kromě toho se můžeme vrátit zpět ve stromu o více úrovní najednou (tzv. *back-jumping*) na místo, kde jsme začali ohodnocovat proměnné v této nové klauzuli. Tím zabráníme opakovanému nalezení "téhož" konfliktu. Více o SAT solverech se dozvíte například v předmětu NAIL094 Decision procedures and SAT/SMT solvers.

Kapitola 4

Metoda analytického tabla

V této kapitole představíme *Metodu analytického tabla*. Jde o syntaktickou proceduru, kterou můžeme použít pro zjištění, zda daný výrok platí v dané teorii, aniž bychom se museli zabývat sémantikou (např. hledat všechny modely, což je nepraktické). Dokážeme si její *korektnost* ('dává správné odpovědi') a *úplnost* ('funguje vždy'), a použijeme ji také k důkazu tzv. *Věty o kompaktnosti* ('vlastnosti nekonečného objektu stačí ukázat pro jeho konečné části').

4.1 Formální dokazovací systémy

Formální dokazovací systém formalizuje 'dokazování' (např. v matematice) jako přesně (algoritmicky) danou syntaktickou proceduru. *Důkaz* faktu, že v teorii T platí výrok φ (neboli $T \models \varphi$) je konečný syntaktický objekt vycházející z axiomů T a výroku φ. Pokud důkaz existuje, lze ho nalézt 'algoritmicky'.[1] Navíc musíme být schopni algoritmicky (a rozumně efektivně) ověřit, že je daný objekt opravdu korektní důkaz.

Existuje-li důkaz, říkáme, že φ je [v daném dokazovacím systému] *dokazatelný* z T, a píšeme $T \vdash \varphi$. Po dokazovacím systému požadujeme dvě vlastnosti:

- *korektnost*: je-li výrok dokazatelný z teorie, je v ní pravdivý ($T \vdash \varphi \Rightarrow T \models \varphi$)
- *úplnost*: je-li výrok pravdivý v teorii, je z ní dokazatelný ($T \models \varphi \Rightarrow T \vdash \varphi$)

(Přičemž korektnost vyžadujeme vždy, ale efektivní dokazovací systém může být praktický, i pokud není úplný, zejména pokud je úplný pro nějakou zajímavou třídu výroků resp. teorií.)

V této kapitole si ukážeme kromě *tablo metody* také *hilbertovský kalkulus*, a v příští kapitole představíme další dokazovací systém, tzv. *rezoluční metodu*.

4.2 Úvod do tablo metody

Po zbytek této kapitoly budeme předpokládat, že máme daný *spočetný* jazyk $\mathbb{P}$. Z toho plyne, že i každá teorie nad $\mathbb{P}$ je spočetná. Nejprve se soustředíme na případ, kdy $T = \emptyset$, tedy dokazujeme, že výrok φ platí *logicky* (je to *tautologie*).

Tablo je olabelovaný strom představující hledání protipříkladu, tj. modelu, ve kterém φ neplatí. Labely na vrcholech, kterým budeme říkat *položky*, sestávají ze symbolu T resp. F

[1] Zde ale musíme být opatrní v případě nekonečné teorie T, jak je zadaná? Algoritmus musí mít efektivní přístup ke všem axiomům.

('True'/'False') následovaného nějakým výrokem ψ a představují předpoklad (požadavek), že v modelu výrok ψ platí resp. neplatí. Do kořene tabla dáme položku Fφ, tj. hledáme model, ve kterém *neplatí* φ. Dále budeme tablo rozvíjet pomocí pravidel pro *redukci* položek. Tato pravidla zajišťují následující invariant:

> Každý model, který se *shoduje* s položkou v kořeni (tj. ve kterém neplatí φ), se musí *shodovat* i s některou větví tabla (tj. splňovat všechny požadavky vyjádřené položkami na této větvi).

Pokud na některé větvi dostaneme položky tvaru Tψ a Fψ (pro totéž ψ), říkáme, že větev *selhala* (je *sporná*) a víme, že žádný model s ní nemůže souhlasit. Pokud selžou všechny větve, víme, že neexistuje žádný model, ve kterém by neplatilo φ, a máme tedy *důkaz*, že φ platí. (Všimněte si, že jde o *důkaz sporem*.)

Pokud nějaká větev neselhala, a je *dokončená*, tj. všechny položky jsou zredukované, víme, že φ neplatí, a budeme z této větve schopni zkonstruovat konkrétní model, ve kterém neplatí.

Příklad 4.2.1. Ukažme si celý postup na dvou příkladech, viz Obrázek 4.2.1.

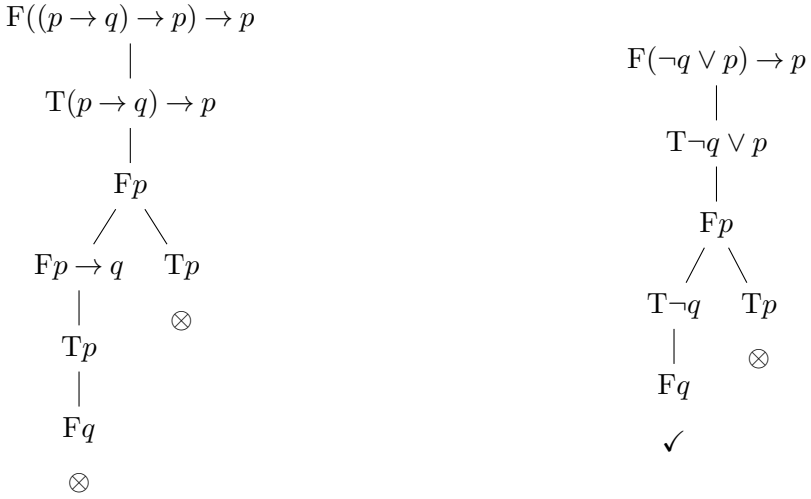
(a) Nejprve sestrojme tablo důkaz výroku $\varphi = ((p \to q) \to p) \to p$. Začneme kořenem s položkou Fφ. Tato položka je tvaru F$\varphi_1 \to \varphi_2$ ('neplatí implikace'), pokud se s ní shoduje nějaký model, musí splňovat T$(p \to q) \to p$ a Fp, připojíme tedy tyto dvě položky. (Ve skutečnosti připojíme *atomické tablo* pro tento případ, viz Tabulka 4.1, kořen tohoto atomického tabla ale vynecháme, abychom zbytečně nezopakovali tutéž položku.) Tím jsme *zredukovali* položku v kořeni.

Pokračujeme položkou T$(p \to q) \to p$, ta je tvaru 'platí implikace', rozvětvíme na dvě větve: model souhlasí s F$(p \to q)$ nebo s Tp (nebo s oběma). Pravá větev *selhala* (je *sporná*), neboť obsahuje položky Tp, Fp, neshoduje se tedy s žádným modelem, označíme ji symbolem $\otimes$. V levé větvi ještě zredukujeme položku F$p \to q$ a také dostaneme spornou větev. Všechny větve jsou sporné, neexistuje tedy žádný protipříklad a máme důkaz výroku φ. Píšeme $\vdash \varphi$.

(b) Nyní sestrojíme tablo s položkou F$(\neg q \vee p) \to p$ v kořeni. Snažíme se tedy najít protipříklad: model, ve kterém neplatí $(\neg q \vee p) \to p$. Nejprve jsme použili atomické tablo pro 'neplatí implikace', a dále redukujeme položku T$\neg q \vee p$ připojením atomického tabla pro 'platí disjunkce'. Pravá větev selhala. V levé větvi ještě zredukujeme T$\neg q$ na Fq (atomické tablo pro 'platí negace') tím dostáváme dokončenou větev, neboť všechny položky už jsme zredukovali. Tato dokončená větev ale není sporná (označíme ji tedy symbolem ✓). To znamená, že protipříklad existuje: máme položky Fp a Fq, kterým odpovídá model $(0, 0)$, ve kterém opravdu $(\neg q \vee p) \to p$ neplatí.

V následující sekci celý postup zformalizujeme a vysvětlíme, co dělat, když chceme dokazovat ne v logice, ale v nějaké teorii T (spoiler alert: při konstrukci připojujeme položky Tα pro axiomy $\alpha \in T$). Také si ukážeme příklad s nekonečnou teorií, kde *dokončená* větev někdy musí být nekonečná.

Ve zbytku této sekce představíme všechna *atomická tabla* potřebná při konstrukci, a také formalizujeme pojem *stromu*.



Obrázek 4.1: Příklady tabel. (a) Tablo důkaz výroku $((p \to q) \to p) \to p$. (b) Tablo pro výrok $(\neg q \vee p) \to p$. Levá větev dává protipříklad, model $(0, 0)$ ve kterém výrok neplatí.

4.2.1 Atomická tabla

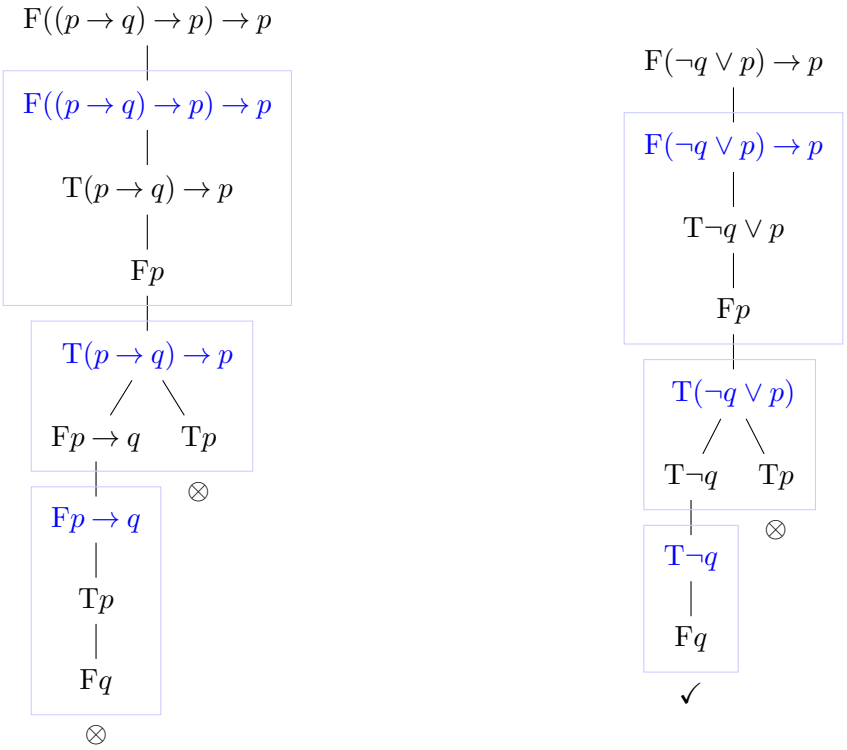
Atomická tabla představují pravidla, pomocí kterých redukujeme položky. Pro každou logickou spojku a každý ze dvou příznaků T/ F máme jedno atomické tablo, znázorněné v Tabulce 4.1.

	$\neg$	$\wedge$	$\vee$	$\to$	$\leftrightarrow$
True	$\mathrm{T}\neg\varphi$ $\mathrm{F}\varphi$	$\mathrm{T}\varphi \wedge \psi$ $\mathrm{T}\varphi$ $\mathrm{T}\psi$	$\mathrm{T}\varphi \vee \psi$ $\mathrm{T}\varphi \quad \mathrm{T}\psi$	$\mathrm{T}\varphi \to \psi$ $\mathrm{F}\varphi \quad \mathrm{T}\psi$	$\mathrm{T}\varphi \leftrightarrow \psi$ $\mathrm{T}\varphi \quad \mathrm{F}\varphi$ $\mathrm{T}\psi \quad \mathrm{F}\psi$
False	$\mathrm{F}\neg\varphi$ $\mathrm{T}\varphi$	$\mathrm{F}\varphi \wedge \psi$ $\mathrm{F}\varphi \quad \mathrm{F}\psi$	$\mathrm{F}\varphi \vee \psi$ $\mathrm{F}\varphi$ $\mathrm{F}\psi$	$\mathrm{F}\varphi \to \psi$ $\mathrm{T}\varphi$ $\mathrm{F}\psi$	$\mathrm{F}\varphi \leftrightarrow \psi$ $\mathrm{T}\varphi \quad \mathrm{F}\varphi$ $\mathrm{F}\psi \quad \mathrm{T}\psi$

Tabulka 4.1: Atomická tabla

Tabla z Příkladu 4.2.1 jsou zkonstruovaná postupným připojováním atomických tabel, viz Obrázek 4.2.1. Kořeny atomických tabel jsou označené modře, zavedeme konvenci, že je nebudeme zakreslovat.

Cvičení 4.1. Pokuste se zkonstruovat tablo s položkou $\mathrm{F}((\neg p \wedge \neg q) \vee p) \to (\neg p \wedge \neg q)$ v kořeni a také tablo s položkou $\mathrm{T}(p \to q) \leftrightarrow (p \wedge \neg q)$. Při konstrukci používejte jen atomická tabla (zkontrolujte, zda vaše konstrukce souhlasí s definicí tabla z následující sekce). Rozmyslete si, co tato tabla říkají o výrocích ve svých kořenech.



Obrázek 4.2: Konstrukce tabel z Příkladu 4.2.1.

Cvičení 4.2. Ověřte, že všechna atomická tabla splňují invariant: shoduje-li se model s položkou v kořeni, shoduje se s některou z větví.

Cvičení 4.3. Navrhněte atomická tabla pro logické spojky NAND, NOR, XOR, IFTE.

4.2.2 O stromech

Než se pustíme do formální definice a důkazů, specifikujme, co myslíme pojmem strom. V teorii grafů bychom stromem nazvali souvislý graf bez cyklů, naše stromy jsou ale zakořeněné, uspořádané (tzv. pravolevým uspořádáním množiny synů každého vrcholu), a označkované. A mohou, často i budou, nekonečné. Formálně:

Definice 4.2.2 (Strom).

- *Strom* je neprázdná množina T s částečným uspořádáním $<_T$, které má (jediný) minimální prvek (*kořen*) a ve kterém je množina předků libovolného vrcholu *dobře uspořádaná*.[2]
- *Větev* stromu T je maximální[3] lineárně uspořádaná podmnožina T.
- *Uspořádaný strom* je strom T spolu s lineárním uspořádáním $<_L$ množiny synů každého vrcholu. Uspořádání synů budeme říkat *pravolevé* zatímco uspořádání $<_T$ je *stromové*.
- *Označkovaný strom* je strom spolu se značkovací funkcí label: $V(T) \to$ Labels.

[2] Tj. každá její neprázdná podmnožina má nejmenší prvek. (Tím zakážeme nekonečné klesající řetězce předků.)

[3] Tj. nelze do ní přidat další vrcholy stromu.

Budeme používat standardní terminologii o stromech, např. budeme mluvit o *n-té úrovni stromu*, nebo o *hloubce* stromu (ta je nekonečná, právě když máme nekonečnou větev). V jedné větě, kterou si níže dokážeme, budeme potřebovat následující slavné tvrzení, které je důsledkem axiomu výběru.

Lemma 4.2.3 (Königovo lemma)**.** *Nekonečný, konečně větvící strom má nekonečnou větev.*

(Strom je *konečně větvící*, pokud má každý vrchol konečně mnoho synů.)

4.3 Tablo důkaz

Nyní uvedeme formální definici tabla. Do definice přidáme také teorii T, jejíž axiomy můžeme při konstrukci připojovat s příznakem T ("true"). Připomeňme, že *položka* je nápis Tφ nebo Fφ, kde φ je nějaký výrok.

Definice 4.3.1 (Tablo)**.** *Konečné tablo z teorie T* je uspořádaný, položkami označkovaný strom zkonstruovaný aplikací konečně mnoha následujících pravidel:

- jednoprvkový strom označkovaný libovolnou položkou je tablo z teorie T,
- pro libovolnou položkou P na libovolné větvi V, můžeme na konec větve V připojit atomické tablo pro položku P,
- na konec libovolné větve můžeme připojit položku Tα pro libovolný axiom teorie $\alpha \in T$.

Tablo z teorie T je buď konečné, nebo i *nekonečné*: v tom případě vzniklo ve spočetně mnoha krocích. Můžeme ho formálně vyjádřit jako sjednocení $\tau = \bigcup_{i \geq 0} \tau_i$, kde τ_i jsou konečná tabla z T, τ_0 je jednoprvkové tablo, a τ_{i+1} vzniklo z τ_i v jednom kroku.[4]

Tablo *pro položku P* je tablo, které má položku P v kořeni.

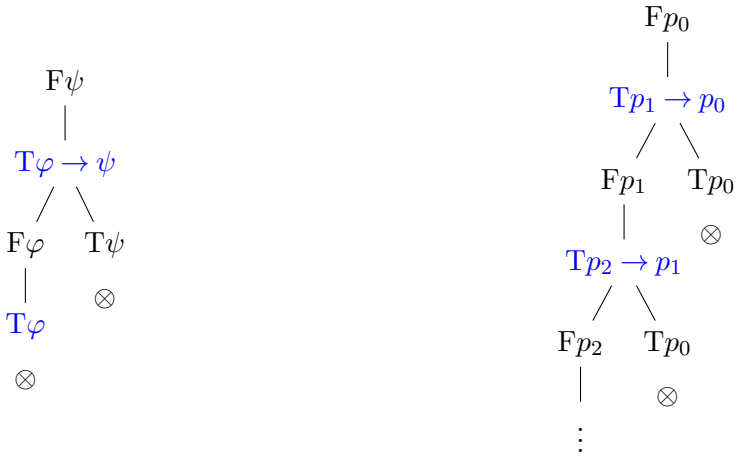
Připomeňme konvenci, že kořen atomického tabla nebudeme zapisovat (neboť vrchol s položkou P už v tablu je). V definici neurčujeme, v jakém pořadí provádět jednotlivé kroky, později ale specifikujeme konkrétní postup konstrukce (algoritmus), kterému budeme říkat *systematické tablo*.

Abychom získali dokazovací systém, zbývá definovat pojem *tablo důkazu* (a související pojmy). Připomeňme ještě jednou, že jde o důkaz sporem, tedy předpokládáme, že výrok neplatí, a najdeme spor(né tablo):

Definice 4.3.2 (Tablo důkaz)**.** *Tablo důkaz* výroku φ z teorie T je *sporné* tablo z teorie T s položkou Fφ v kořeni. Pokud existuje, je *φ (tablo) dokazatelný* z T, píšeme $T \vdash \varphi$. (Definujme také *tablo zamítnutí* jako sporné tablo s Tφ v kořeni. Pokud existuje, je *φ (tablo) zamítnutelný* z T, tj. platí $T \vdash \neg\varphi$.)

- Tablo je *sporné*, pokud je každá jeho větev sporná.
- Větev je *sporná*, pokud obsahuje položky Tψ a Fψ pro nějaký výrok ψ, jinak je *bezesporná*.
- Tablo je *dokončené*, pokud je každá jeho větev dokončená.

[4] Sjednocení proto, že v jednotlivých krocích přidáváme do tabla nové vrcholy, τ_i je tedy podstromem τ_{i+1}.



Obrázek 4.3: Tabla z Příkladu 4.3.3. Položky vycházející z axiomů jsou označeny modře.

- Větev je *dokončená*, pokud
  - je sporná, nebo
  - je každá její položka na této větvi *redukovaná* a zároveň obsahuje položku Tα pro každý axiom $\alpha \in T$.
- Položka P je *redukovaná* na větvi V procházející touto položkou, pokud
  - je tvaru Tp resp. Fp pro nějakou výrokovou proměnnou $p \in \mathbb{P}$, nebo
  - vyskytuje se na V jako kořen atomického tabla[5] (tj., typicky, při konstrukci tabla již došlo k jejímu rozvoji na V).

Příklad 4.3.3. Ukážeme si dva příklady. Tabla jsou znázorněná na Obrázku 4.3.

(a) Tablo důkaz výroku ψ z teorie $T = \{\varphi, \varphi \to \psi\}$, tj. $T \vdash \psi$ (kde φ, ψ jsou nějaké pevně dané výroky). Tomuto faktu se říká *Věta o dedukci*.

(b) Dokončené tablo pro výrok p_0 z teorie $T = \{p_{n+1} \to p_n \mid n \in \mathbb{N}\}$. Nejlevější větev je bezesporná dokončená. Obsahuje položky T$p_{i+1} \to p_i$ a Fp_i pro všechna $i \in \mathbb{N}$. Shoduje se tedy s modelem $v = (0, 0, \dots)$, tj. $v : \mathbb{P} \to \{0, 1\}$ kde $v(p_i) = 0$ pro všechna i.

Cvičení 4.4. Vraťme se k tablům z Cvičení 4.1. Jde o tablo důkazy nebo zamítnutí (z teorie $T = \emptyset$)? Které položky na kterých větvích jsou redukované? Které větve jsou sporné, které jsou dokončené?

4.4 Konečnost a systematičnost důkazů

V této sekci dokážeme, že pokud existuje tablo důkaz, existuje vždy také *konečný* tablo důkaz. Představíme také algoritmus, kterým nějaký tablo důkaz můžeme vždy najít, pro důkaz tohoto faktu ale budeme potřebovat Věty o korektnosti a úplnosti z následující sekce. Prozatím ukážeme, že tento algoritmus nám umožní vždy sestrojit dokončené tablo.

[5] Byť podle konvence tento kořen nezapisujeme.

Všimněte si, že při redukci položky přidáváme do tabla pouze položky obsahující kratší výroky. Pokud tedy máme konečnou teorii, a neděláme zbytečné kroky (například nepřidáváme opakovaně tentýž axiom, nebo totéž atomické tablo), je snadné sestrojit dokončené tablo, které bude konečné.

Je-li teorie T nekonečná, musíme ale být opatrnější. Mohli bychom nekonečně dlouho konstruovat tablo, a přitom se nikdy nedostat k redukci určité položky, nebo nikdy nepoužít některý z axiomů. Definujeme tedy konkrétní algoritmus pro konstrukci tabla, výsledku budeme říkat *systematické tablo*. Myšlenka konstrukce je jednoduchá: střídáme krok redukce položky (zároveň na všech bezesporných větvích, které jí procházejí) a krokem použití axiomu. Položky procházíme po úrovních, a v rámci úrovně v pravolevém uspořádání. A axiomy teorie ve zvoleném očíslování.

Definice 4.4.1. Mějme položku R a (konečnou nebo nekonečnou[6]) teorii $T = \{\alpha_1, \alpha_2, \dots\}$. *Systematické tablo* z teorie T pro položku R je tablo $\tau = \bigcup_{i \geq 0} \tau_i$, kde τ_0 je jednoprvkové tablo s položkou R, a pro každé $i \geq 0$:

- Nechť P je nejlevější položka v co nejmenší úrovni, která není redukovaná na nějaké bezesporné větvi procházející P. Definujeme nejprve tablo τ_i' jako tablo vzniklé z τ_i připojením atomického tabla pro P na každou bezespornou větev procházející P. (Pokud taková položka neexistuje, potom $\tau_i' = \tau_i$.)
- Následně, τ_{i+1} je tablo vzniklé z τ_i' připojením $\mathrm{T}\alpha_{i+1}$ na každou bezespornou větev τ_i'. To v případě, že $i < |T|$, jinak (je-li T konečná a už jsme použili všechny axiomy) tento krok přeskočíme a definujeme $\tau_{i+1} = \tau_i'$.

Lemma 4.4.2. *Systematické tablo je dokončené.*

Důkaz. Ukážeme, že každá větev je dokončená. Sporné větve jsou dokončené. Bezesporné větve obsahují položky $\mathrm{T}\alpha_i$ (ty jsme připojili v i-tém kroku) a každá položka na nich je redukovaná. Vskutku, kdyby P byla neredukovaná na bezesporné větvi V, přišla by na ni v nějakém kroku řada, neboť v úrovních nad P a vlevo od P existuje jen konečně mnoho položek. (Používáme zjevného faktu, že každý prefix bezesporné větve je také bezesporná větev, tedy během konstrukce V nikdy není sporná.) □

Nyní se vraťme k otázce konečnosti důkazů:

Věta 4.4.3 (Konečnost sporu)**.** *Je-li $\tau = \bigcup_{i \geq 0} \tau_i$ sporné tablo, potom existuje $n \in \mathbb{N}$ takové, že τ_n je sporné konečné tablo.*

Důkaz. Uvažme množinu S všech vrcholů stromu τ, které nad sebou (ve stromovém uspořádání) neobsahují spor, tj. dvojici položek $\mathrm{T}\psi$, $\mathrm{F}\psi$.

Kdyby množina S byla nekonečná, podle Königova lemmatu použitého na podstrom τ na množině S bychom měli nekonečnou, bezespornou větev v S. To by ale znamenalo, že máme i bezespornou větev v τ, což je ve sporu s tím, že τ je sporné. (Podrobněji: Větev na S by byla podvětví nějaké větve V v τ, která je sporná, tj. obsahuje nějakou (konkrétní) spornou dvojici položek, která ale existuje už v nějakém konečném prefixu V.)

Množina S je tedy konečná. To znamená, že existuje $d \in \mathbb{N}$ takové, že celá S leží v hloubce nejvýše d. Každý vrchol na úrovni $d+1$ má tedy nad sebou spor. Zvolme n tak, že τ_n už obsahuje všechny vrcholy τ z prvních $d+1$ úrovní: každá větev τ_n je tedy sporná. □

[6] Připomeňme, že T je spočetná, neboť jazyk je (v celé kapitole) spočetný.

Důsledek 4.4.4. *Pokud při konstrukci tabla nikdy neprodlužujeme sporné větve, např. pro systematické tablo, potom sporné tablo je konečné.*

Důkaz. Použijeme Větu 4.4.3, máme $\tau = \tau_n$ neboť sporné tablo už neměníme. □

Důsledek 4.4.5 (Konečnost důkazů)**.** *Pokud $T \vdash \varphi$, potom existuje i* konečný *tablo důkaz φ z T.*

Důkaz. Snadno plyne z Důsledku 4.4.4: stačí při konstrukci τ ignorovat kroky, které by prodloužily spornou větev. □

Vyslovíme zde také následující důsledek. Dokážeme ho ale až v příští sekci.

Důsledek 4.4.6 (Systematičnost důkazů)**.** *Pokud $T \vdash \varphi$, potom systematické tablo je (konečným) tablo důkazem φ z T.*

K důkazu budeme potřebovat dvě fakta: pokud je φ dokazatelná z T, potom v T platí (Věta o korektnosti), tj. nemůže existovat protipříklad. A dále pokud by systematické tablo mělo bezespornou větev, znamenalo by to, že existuje protipříklad (to je klíčem k Větě o úplnosti).

4.5 Korektnost a úplnost

V této sekci dokážeme, že je tablo metoda *korektní* a *úplný* dokazovací systém, tj. že $T \vdash \varphi$ platí právě když $T \models \varphi$.

4.5.1 Věta o korektnosti

Řekneme, model v se *shoduje* s položkou P, pokud $P = \mathrm{T}\varphi$ a $v \models \varphi$, nebo $P = \mathrm{F}\varphi$ a $v \not\models \varphi$. Dále v se shoduje s větví V, pokud se shoduje s každou položkou na této větvi.

Jak už jsme zmínili, design atomických tabel zaručuje, že shoduje-li se model s položkou v kořeni tabla, shoduje se s některou větví. Není těžké indukcí podle konstrukce tabla ukázat následující lemma:

Lemma 4.5.1. *Shoduje-li se model teorie T s položkou v kořeni tabla z teorie T, potom se shoduje s některou větví.*

Důkaz. Mějme tablo $\tau = \bigcup_{i \geq 0} \tau_i$ z teorie T a model $v \in \mathrm{M}(T)$ shodující se s kořenem τ, tedy s (jednoprvkovou) větví V_0 v (jednoprvkovém) τ_0.

Indukcí podle i (podle kroků v při konstrukci tabla) najdeme posloupnost $V_0 \subseteq V_1 \subseteq \ldots$ takovou, že V_i je větev v tablu τ_i shodující se s modelem v, a V_{i+1} je prodloužením V_i. Požadovaná větev tabla τ je potom $V = \bigcup_{i \geq 0} V_i$.

- Pokud τ_{i+1} vzniklo z τ_i bez prodloužení větve V_i, definujeme $V_{i+1} = V_i$.
- Pokud τ_{i+1} vzniklo z τ_i připojením položky $\mathrm{T}\alpha$ (pro nějaký axiom $\alpha \in T$) na konec větve V_i, definujeme V_{i+1} jako tuto prodlouženou větev. Protože v je model T, platí v něm axiom α, tedy shoduje se i s novou položkou $\mathrm{T}\alpha$.

- Nechť τ_{i+1} vzniklo z τ_i připojením atomického tabla pro nějakou položku P na konec větve V_i. Protože se model v shoduje s položkou P (která leží už na větvi V_i), shoduje se i s kořenem připojeného atomického tabla, a proto se shoduje i s některou z jeho větví. (Tuto vlastnost snadno ověříme pro všechna atomická tabla.) Definujeme V_{i+1} jako prodloužení V_i o tuto větev atomického tabla.[7]

□

Nyní už můžeme dokázat Větu o korektnosti. Zkráceně řečeno, pokud by existoval důkaz a zároveň protipříklad, protipříklad by se musel shodovat s některou větví důkazu, ty jsou ale všechny sporné.

Věta 4.5.2 (O korektnosti)**.** *Je-li výrok φ tablo dokazatelný z teorie T, potom je φ pravdivý v T, tj. $T \vdash \varphi \;\Rightarrow\; T \models \varphi$.*

Důkaz. Dokážeme sporem. Předpokládejme, že φ v T neplatí, tj. existuje protipříklad: model $v \in \mathrm{M}(T)$, ve kterém φ neplatí.

Protože je φ dokazatelná z T, existuje tablo důkaz φ z T, což je sporné tablo z T s položkou $\mathrm{F}\varphi$ v kořeni. Model v se shoduje s položkou $\mathrm{F}\varphi$, tedy podle Lemmatu 4.5.1 se shoduje s nějakou větví V. Všechny větve jsou ale sporné, včetně V. Takže V obsahuje položky $\mathrm{T}\psi$ a $\mathrm{F}\psi$ (pro nějaký výrok ψ), a model v se s těmito položkami shoduje. Máme tedy $v \models \psi$ a zároveň $v \not\models \psi$, což je spor.

□

4.5.2 Věta o úplnosti

Ukážeme, že pokud selže dokazování, tj. pokud dostaneme *bezespornou* větev v *dokončeném* tablu z teorie T pro položku $\mathrm{F}\varphi$, potom tato větev poskytuje protipříklad: model teorie T, který se shoduje s položkou $\mathrm{F}\varphi$ v kořeni tabla, tj. neplatí v něm φ. Takových modelů může být více, definujeme proto jeden konkrétní:

Definice 4.5.3 (Kanonický model)**.** Je-li V bezesporná větev dokončeného tabla, potom *kanonický model* pro V je model definovaný předpisem (pro $p \in \mathbb{P}$):

$$v(p) = \begin{cases} 1 \text{ pokud se na } V \text{ vyskytuje položka } \mathrm{T}p, \\ 0 \text{ jinak.} \end{cases}$$

Lemma 4.5.4. *Kanonický model pro (bezespornou dokončenou) větev V se shoduje s V.*

Důkaz. Ukážeme, že kanonický model v se shoduje se všemi položkami P na větvi V, a to indukcí podle struktury výroku v položce.[8] Nejprve základ indukce:

- Je-li $P = \mathrm{T}p$ pro nějaký prvovýrok $p \in \mathbb{P}$, máme podle definice $v(p) = 1$; v se s P shoduje.
- Je-li $P = \mathrm{F}p$, potom se na větvi V nemůže vyskytovat položka $\mathrm{T}p$, jinak by V byla sporná. Podle definice máme $v(p) = 0$ a v se s P opět shoduje.

[7] Resp. o libovolnou takovou větev: model v se může shodovat s více větvemi atomického tabla.
[8] Připomeňme, že to znamená indukci podle hloubky stromu výroku.

Nyní indukční krok. Rozebereme dva případy, ostatní se dokáží obdobně.

- Nechť $P = \mathrm{T}\varphi \wedge \psi$. Protože je V dokončená větev, je na ní položka P redukovaná. To znamená, že se na V vyskytují i položky $\mathrm{T}\varphi$ a $\mathrm{T}\psi$. Podle indukčního předpokladu se s nimi model v shoduje, tedy $v \models \varphi$ a $v \models \psi$. Takže platí i $v \models \varphi \wedge \psi$ a v se shoduje s P.
- Nechť $P = \mathrm{F}\varphi \wedge \psi$. Protože je P na V redukovaná, vyskytuje se na V položka $\mathrm{F}\varphi$ nebo položka $\mathrm{F}\psi$. Platí tedy $v \not\models \varphi$ nebo $v \not\models \psi$, z čehož plyne $v \not\models \varphi \wedge \psi$ a v se shoduje s P.

□

Věta 4.5.5 (O úplnosti)**.** *Je-li výrok φ pravdivý v teorii T, potom je tablo dokazatelný z T, tj. $T \models \varphi \;\Rightarrow\; T \vdash \varphi$.*

Důkaz. Ukážeme, že libovolné *dokončené* (tedy např. i *systematické*) tablo z T s položkou $\mathrm{F}\varphi$ v kořeni je nutně sporné. Důkaz provedeme sporem: kdyby takové tablo nebylo sporné, existovala by v něm bezesporná (dokončená) větev V. Uvažme kanonický model v pro tuto větev. Protože je V dokončená, obsahuje $\mathrm{T}\alpha$ pro všechny axiomy $\alpha \in T$. Model v se podle Lemmatu 4.5.4 shoduje se všemi položkami na V, splňuje tedy všechny axiomy a máme $v \models T$. Protože se ale v shoduje i s položkou $\mathrm{F}\varphi$ v kořeni, máme $v \not\models \varphi$, což znamená, že $T \not\models \varphi$, spor. Tablo tedy muselo být sporné, tj. být tablo důkazem φ z T. □

Důkaz Důsledku 4.4.6. Z předchozího důkazu také dostáváme 'systematičnost důkazů', tj. že důkaz můžeme vždy hledat konstrukcí systematického tabla: Pokud $T \models \varphi$, tak je i systematické tablo pro položku $\mathrm{F}\varphi$ nutně sporné, a je tedy tablo důkazem φ z T. □

Cvičení 4.5. Ověřte zbývající případy v důkazu Lemmatu 4.5.4.

Cvičení 4.6. Popište, jak vypadají *všechny* modely shodující se s danou bezespornou dokončenou větví.

Cvičení 4.7. Navrhněte postup, kterým můžeme za použití tablo metody najít všechny modely dané teorie T.

4.6 Důsledky korektnosti a úplnosti

Věty o korektnosti a úplnosti dohromady říkají, že *dokazatelnost* je totéž, co *platnost*. To nám umožňuje zformulovat syntaktické analogie sémantických pojmů a vlastností.

Analogií *důsledků* jsou *teorémy* teorie T:

$$\mathrm{Thm}_{\mathbb{P}}(T) = \{\varphi \in \mathrm{VF}_{\mathbb{P}} \mid T \vdash \varphi\}$$

Důsledek 4.6.1 (Dokazatelnost = platnost)**.** *Pro libovolnou teorii T a výroky φ, ψ platí:*

- *$T \vdash \varphi$ právě když $T \models \varphi$*
- $\mathrm{Thm}_{\mathbb{P}}(T) = \mathrm{Csq}_{\mathbb{P}}(T)$

Důkaz. Plyne okamžitě z Věty o korektnosti a z Věty o úplnosti. □

Ve všech definicích a větách můžeme tedy nahradit pojem '*platnost*' pojmem '*dokazatelnost*' (tj. symbol '$\models$' symbolem '$\vdash$') a pojem '*důsledek*' pojmem '*teorém*'. Například:

- Teorie je *sporná*, jestliže je v ní dokazatelný spor (tj. $T \vdash \bot$).
- Teorie je *kompletní*, jestliže pro každý výrok φ je buď $T \vdash \varphi$ nebo $T \vdash \neg\varphi$ (ale ne obojí, jinak by byla sporná).

Uveďme ještě jeden snadný důsledek:

Věta 4.6.2 (O dedukci)**.** *Pro teorii T a výroky φ, ψ platí: $T, \varphi \vdash \psi$ právě když $T \vdash \varphi \to \psi$.*

Důkaz. Stačí dokázat $T, \varphi \models \psi \Leftrightarrow T \models \varphi \to \psi$, což je snadné. □

Cvičení 4.8. Dokažte Větu o dedukci přímo, pomocí transformace tablo důkazů.

4.7 Věta o kompaktnosti

Důležitým důsledkem vět o korektnosti a úplnosti je také tzv. *Věta o kompaktnosti*.[9] Tento princip umožňuje převádět tvrzení o nekonečných objektech/procesech na tvrzení o (všech) jejich konečných částech.

Věta 4.7.1 (O kompaktnosti)**.** *Teorie má model, právě když každá její konečná část má model.*

Důkaz. Každý model teorie T je zjevně modelem každé její části. Druhou implikaci dokážeme nepřímým důkazem: Předpokládejme, že T nemá model, tj. je sporná, a najděme konečnou část $T' \subseteq T$, která je také sporná.

Protože je T sporná, platí $T \vdash \bot$ (zde potřebujeme Větu o úplnosti). Podle Důsledku 4.4.5 potom existuje *konečný* tablo důkaz τ výroku $\bot$ z T. Konstrukce tohoto důkazu má jen konečně mnoho kroků, použili jsme tedy jen konečně mnoho axiomů z T. Definujeme-li $T' = \{\alpha \in T \mid \mathrm{T}\alpha \text{ je položka v tablu } \tau\}$, potom τ je také tablo důkaz sporu z teorie T'. Teorie T' je tedy sporná konečná část T. □

4.7.1 Aplikace kompaktnosti

Následující jednoduchou aplikaci Věty o kompaktnosti můžete chápat jako šablonu, kterou následuje i mnoho dalších, složitějších aplikací této věty.

Důsledek 4.7.2. *Spočetně nekonečný graf je bipartitní, právě když je každý jeho konečný podgraf bipartitní.*

Důkaz. Každý podgraf bipartitního grafu je zjevně také bipartitní. Ukažme opačnou implikaci. Graf je bipartitní, právě když je obarvitelný 2 barvami. Označme barvy $0, 1$.

Sestrojíme výrokovou teorii T v jazyce $\mathbb{P} = \{p_v \mid v \in V(G)\}$, kde hodnota výrokové proměnné p_v reprezentuje barvu vrcholu v.

$$T = \{p_u \leftrightarrow \neg p_v \mid \{u, v\} \in E(G)\}$$

Zřejmě platí, že G je bipartitní, právě když T má model. Podle Věty o kompaktnosti stačí ukázat, že každá konečná část T má model. Vezměme tedy konečnou $T' \subseteq T$. Buď G' podgraf

[9] Slovo *kompaktnost* pochází z kompaktních (tj. omezených a uzavřených) množin v Euklidovských prostorech, ve kterých lze z každé posloupnosti vybrat konvergentní podposloupnost. Můžete si představit posloupnost zvětšujících se konečných částí 'konvergující' k nekonečnému celku.

G indukovaný na množině vrcholů, o kterých se zmiňuje teorie T', tj. $V(G') = \{v \in V(G) \mid p_v \in \mathrm{Var}(T')\}$. Protože je T' konečná, je G' také konečný, a podle předpokladu je 2-obarvitelný. Libovolné 2-obarvení $V(G')$ ale určuje model teorie T'. □

Základem této techniky je popis požadované vlastnosti nekonečného objektu pomocí (nekonečné) výrokové teorie. Dále si všimněte, jak z konečné části teorie sestrojíme konečný podobjekt mající danou vlastnost (v našem případě konečný podgraf, který je bipartitní).

Cvičení 4.9. Zobecněte Důsledek 4.7.2 pro více barev, tj. ukažte, že spočetně nekonečný graf je k-obarvitelný, právě když je každý jeho konečný podgraf k-obarvitelný. (Viz Sekce 1.1.7.)

Cvičení 4.10. Ukažte, že každé částečné uspořádání na spočetné množině lze rozšířit na lineární uspořádání.

Cvičení 4.11. Vyslovte a dokažte 'spočetně nekonečnou' analogii Hallovy věty.

4.8 Hilbertovský kalkulus

Na závěr kapitoly o tablo metodě si pro srovnání ukážeme jiný dokazovací systém, tzv. *hilbertovský deduktivní systém* neboli *hilbertovský kalkulus*. Jde o nejstarší dokazovací systém, modelovaný podle matematických důkazů. Jak uvidíme na příkladě, dokazování je v něm poměrně pracné, hodí se tedy spíše pro teoretické účely. Jde také o korektní a úplný dokazovací systém. (Korektnost ukážeme, úplnost ale necháme bez důkazu.)

Hilbertovský kalkulus používá jen dvě základní logické spojky: negaci a implikaci. (Připomeňme, že ostatní logické spojky z nich lze odvodit.) Systém sestává z logických axiomů daných následujícími *schématy*, a z jednoho *odvozovacího pravidla*, tzv. *modus ponens*:

Definice 4.8.1 (Schémata axiomů v hilbertovském kalkulu)**.** Pro libovolné výroky φ, ψ, χ jsou následující výroky logickými axiomy:

(i) $\varphi \to (\psi \to \varphi)$

(ii) $(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$

(iii) $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$

Všimněte si, že všechny logické axiomy jsou opravdu tautologie. Poznamenejme, že lze zvolit i jiný systém logických axiomů, existuje jich celá řada, viz článek List of Hilbert systems na Wikipedii.

Definice 4.8.2 (Modus ponens)**.** Odvozovací pravidlo *modus ponens* říká, že pokud jsme již dokázali výrok φ a také výrok $\varphi \to \psi$, můžeme odvodit i výrok ψ. Zapisujeme ho následovně:

$$\frac{\varphi, \varphi \to \psi}{\psi}$$

Všimněte si, že modus ponens je *korektní*, tj. platí-li v nějaké teorii $T \models \varphi$ a $T \models \varphi \to \psi$, máme i $T \models \psi$.

Nyní jsme již připraveni definovat *důkaz*. Půjde o konečnou posloupnost výroků, ve které každý nově napsaný výrok je buď axiomem (logickým nebo z teorie, ve které dokazujeme), nebo lze odvodit z předchozích pomocí modus ponens:

Definice 4.8.3 (Hilbertovský důkaz)**.** *Hilbertovský důkaz* výroku φ z teorie T je *konečná* posloupnost výroků $\varphi_0, \ldots, \varphi_n = \varphi$, ve které pro každé $i \leq n$ platí:

- φ_i je logický axiom, nebo
- φ_i je axiom teorie ($\varphi_i \in T$), nebo
- φ_i lze odvodit z nějakých předchozích výroků φ_j, φ_k (kde $j, k < i$) pomocí modus ponens.

Existuje-li hilbertovský důkaz, říkáme, že je φ *(hilbertovsky) dokazatelný*, a píšeme $T \vdash_H \varphi$.

Pojem hilbertovského důkazu si ilustrujeme na jednoduchém příkladě:

Příklad 4.8.4. Ukažme, že pro teorii $T = \{\neg\varphi\}$ a pro libovolný výrok ψ platí $T \vdash_H \varphi \to \psi$. Hilbertovským důkazem je následující posloupnost výroků:

1. $\neg\varphi$	*axiom teorie*
2. $\neg\varphi \to (\neg\psi \to \neg\varphi)$	*logický axiom dle (i)*
3. $\neg\psi \to \neg\varphi$	*modus ponens na 1. a 2.*
4. $(\neg\psi \to \neg\varphi) \to (\varphi \to \psi)$	*logický axiom dle (iii)*
5. $\varphi \to \psi$	*modus ponens na 3. a 4.*

Jak jsme již zmínili, hilbertovský kalkulus je korektní a úplný dokazovací systém.

Věta 4.8.5 (O korektnosti hilbertovského kalkulu)**.** *Pro každou teorii T a výrok φ platí:*

$$T \vdash_H \varphi \;\Rightarrow\; T \models \varphi$$

Důkaz. Indukcí dle indexu i ukážeme, že každý výrok φ_i z důkazu (tedy i $\varphi_n = \varphi$) platí v T.

Je-li φ_i logický axiom, $T \models \varphi_i$ platí protože logické axiomy jsou tautologie. Je-li $\varphi_i \in T$, také jistě platí $T \models \varphi_i$. Získáme-li φ_i pomocí modus ponens z φ_j a $\varphi_k = \varphi_j \to \varphi_i$ (pro nějaká $j, k < i$), víme z indukčního předpokladu, že platí $T \models \varphi_j$ a $T \models \varphi_j \to \varphi_i$. Potom ale z korektnosti modus ponens platí i $T \models \varphi_i$. □

Pro úplnost ještě vyslovme úplnost, důkaz ale neuvedeme.

Věta 4.8.6 (O úplnosti hilbertovského kalkulu)**.** *Pro každou teorii T a výrok φ platí:*

$$T \models \varphi \;\Rightarrow\; T \vdash_H \varphi$$

Kapitola 5

Rezoluční metoda

V této kapitole představíme jiný dokazovací systém, vhodnější pro praktické aplikace, tzv. *rezoluční metodu*. Tato metoda je základem např. *logického programování* nebo systémů *automatického dokazování* a *softwarové verifikace*. V této kapitole se omezíme na rezoluční metodu ve výrokové logice, ale později, v Kapitole **??**, si ukážeme koncept *unifikace*, který umožňuje hledat rezoluční důkazy v logice predikátové.

Rezoluční metoda pracuje s výroky v *konjunktivní normální formě (CNF)*. Připomeňme, že každý výrok lze převést do CNF. Tento převod je v nejhorším případě v exponenciálním čase (dokonce existují výroky jejichž nejkratší CNF ekvivalent je exponenciálně delší), v praxi to ale není problém.

Podobně jako tablo metoda je založena na důkazu sporem, tj. přidáme k teorii, ve které dokazujeme, *negaci* výroku, který chceme dokázat (obojí převedené do CNF), a ukážeme, že to vede ke sporu.

K hledání sporu používá rezoluční metoda jediné inferenční pravidlo, tzv. *rezoluční pravidlo*. To je speciálním případem *pravidla řezu*, které říká: *"z výroků* $\varphi \vee \psi$ *a* $\neg\varphi \vee \chi$ *lze odvodit výrok* $\psi \vee \chi$*,"* píšeme:

$$\frac{\varphi \vee \psi, \neg\varphi \vee \chi}{\psi \vee \chi}$$

V *rezolučním pravidle*, které si ukážeme za chvíli, bude φ *literál*, a ψ, χ budou klauzule.

Cvičení 5.1. Rozmyslete si, že pravidlo řezu je *korektní*. (Co to znamená, a proč to platí?)

5.1 Množinová reprezentace

Nejprve představíme úspornější zápis CNF výroků, tzv. *množinový zápis*. Bylo by totiž nepraktické zapisovat výroky včetně závorek a logických symbolů.

- Připomeňme, že *Literál* ℓ je prvovýrok nebo negace prvovýroku a že $\bar{\ell}$ označuje *opačný literál* k ℓ.
- *Klauzule* C je konečná množina literálů. *Prázdnou klauzuli*, která není nikdy splněna,[1] označíme $\square$.

[1] Reprezentuje disjunkci prázdné množiny literálů, žádný z disjunktů tedy není splněný.

- *(CNF) formule* S je (konečná, nebo i nekonečná) množina klauzulí. *Prázdná formule* $\emptyset$ je vždy splněna.[2]

Poznámka 5.1.1. Všimněte si, že *CNF formule* může být i *nekonečná* množina klauzulí. Pokud tedy převádíme nekonečnou výrokovou teorii do CNF, zapíšeme v množinové reprezentaci všech nekonečně mnoho klauzulí jako prvky jediné formule (množiny). V praktických aplikacích je samozřejmě formule (téměř vždy) konečná.

V množinové reprezentaci odpovídají modely množinám literálů, které obsahují pro každou výrokovou proměnnou p právě jeden z literálů $p, \neg p$:

- *(Částečné) ohodnocení* $\mathcal{V}$ je libovolná množina literálů, která je *konzistentní*, tj. neobsahuje dvojici opačných literálů.
- Ohodnocení je *úplné*, pokud obsahuje pozitivní nebo negativní literál pro každou výrokovou proměnnou.
- Ohodnocení $\mathcal{V}$ *splňuje* formuli S, píšeme $\mathcal{V} \models S$, pokud $\mathcal{V}$ obsahuje nějaký literál z každé klauzule v S, tj.:
$$\mathcal{V} \cap C \neq \emptyset \text{ pro každou } C \in S$$

Příklad 5.1.2. Výrok $\varphi = (\neg p_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_3 \vee \neg p_4) \wedge (\neg p_4 \vee \neg p_5) \wedge p_4$ zapíšeme v množinové reprezentaci takto:

$$S = \{\{\neg p_1, p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_3, \neg p_4\}, \{\neg p_4, \neg p_5\}, \{p_4\}\}$$

Ohodnocení $\mathcal{V} = \{\neg p_1, \neg p_3, p_4, \neg p_5\}$ splňuje S, píšeme $\mathcal{V} \models S$. Není úplné, ale můžeme ho rozšířit libovolným literálem pro p_2: platí $\mathcal{V} \cup \{p_2\} \models S$ i $\mathcal{V} \cup \{\neg p_2\} \models S$. Tato dvě úplná ohodnocení odpovídají modelům $(0, 1, 0, 1, 0)$ a $(0, 0, 0, 1, 0)$.

5.2 Rezoluční důkaz

Nejprve definujeme jeden krok inference v rezolučním důkazu, tzv. *rezoluční pravidlo*, které aplikujeme na dvojici klauzulí; jeho výsledkem je klauzule, které říkáme *rezolventa*, a která je logickým důsledkem původní dvojice klauzulí:

Definice 5.2.1 (Rezoluční pravidlo)**.** Mějme klauzule C_1 a C_2 a literál ℓ takový, že $\ell \in C_1$ a $\bar{\ell} \in C_2$. Potom *rezolventa* klauzulí C_1 a C_2 *přes literál* ℓ je klauzule

$$C = (C_1 \setminus \{\ell\}) \cup (C_2 \setminus \{\bar{\ell}\}).$$

Z první klauzule tedy odstraníme literál ℓ a z druhé literál $\bar{\ell}$ (které tam musely být!) a všechny zbylé literály sjednotíme do výsledné rezolventy. S pomocí symbolu $\dot\cup$ pro disjunktní sjednocení bychom také mohli psát:

$$C_1' \cup C_2' \text{ je rezolventou klauzulí } C_1' \dot\cup \{\ell\} \text{ a } C_2' \dot\cup \{\bar{\ell}\}$$

Příklad 5.2.2. Z klauzulí $C_1 = \{\neg q, r\}$ a $C_2 = \{\neg p, \neg q, \neg r\}$ lze odvodit rezolventu $\{\neg p, \neg q\}$ přes literál r. Z klauzulí $\{p, q\}$ a $\{\neg p, \neg q\}$ lze odvodit $\{p, \neg p\}$ přes literál q nebo $\{q, \neg q\}$ přes literál p (obojí jsou ale tautologie).[3]

[2] Reprezentuje konjunkci prázdné množiny klauzulí, všechny klauzule v S jsou tedy splněny.

[3] Nelze ale odvodit $\square$ 'rezolucí přes p a q najednou' (což je častá chyba). Všimněte si, že $\{\{p, q\}, \{\neg p, \neg q\}\}$ není nesplnitelná, např. $(1, 0)$ je modelem.

Pozorování 5.2.3 (Korektnost rezolučního pravidla)**.** *Rezoluční pravidlo je korektní, tj. pro libovolné ohodnocení $\mathcal{V}$ platí:*

$$\textit{Pokud } \mathcal{V} \models C_1 \textit{ a } \mathcal{V} \models C_2\textit{, potom } \mathcal{V} \models C.$$

Rezoluční důkaz definujeme podobně jako v Hilbertově kalkulu jako konečnou posloupnost klauzulí, kde je zaručena platnost každé klauzule v této posloupnosti: v každém kroku můžeme buď napsat 'axiom' (klauzuli z S), nebo rezolventu nějakých dvou už napsaných klauzulí.

Definice 5.2.4 (Rezoluční důkaz)**.** *Rezoluční důkaz (odvození)* klauzule C z CNF formule S je *konečná* posloupnost klauzulí $C_0, C_1, \ldots, C_n = C$ taková, že pro každé i buď $C_i \in S$ 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* CNF formule S je rezoluční důkaz $\square$ z S, v tom případě je S *(rezolucí) zamítnutelná.*

Příklad 5.2.5. CNF formule $S = \{\{p, \neg q, r\}, \{p, \neg r\}, \{\neg p, r\}, \{\neg p, \neg r\}, \{q, r\}\}$ je (rezolucí) zamítnutelná, jedno z možných zamítnutí je:

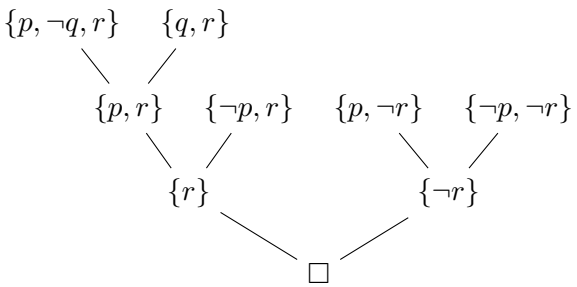
$$\{p, \neg q, r\}, \{q, r\}, \{p, r\}, \{\neg p, r\}, \{r\}, \{p, \neg r\}, \{\neg p, \neg r\}, \{\neg r\}, \square$$

Rezoluční důkaz má přirozenou stromovou strukturu: v listech jsou axiomy a vnitřní vrcholy představují jednotlivé rezoluční kroky.

Definice 5.2.6 (Rezoluční strom)**.** Rezoluční strom klauzule C z CNF formule S je *konečný* binární strom s vrcholy označenými klauzulemi, kde

- v kořeni je C,
- v listech jsou klauzule z S,
- v každém vnitřním vrcholu je rezolventa klauzulí ze synů tohoto vrcholu.

Příklad 5.2.7. Rezoluční strom prázdné klauzule $\square$ z CNF formule S z Příkladu 5.2.5 je:



Je snadné ukázat následující pozorování, indukcí podle hloubky stromu a délky rezolučního důkazu:

Pozorování 5.2.8. *Klauzule C má rezoluční strom z CNF formule S, právě když $S \vdash_R C$.*

Každému rezolučnímu důkazu odpovídá jednoznačný rezoluční strom. Naopak, z jednoho rezolučního stromu můžeme získat více rezolučních důkazů: jsou dané libovolnou procházkou po vrcholech stromu, při které navštívíme vnitřní vrchol až poté, co jsme navštívili oba jeho syny.

Zaveďme ještě jeden pojem, tzv. *rezoluční uzávěr*, který obsahuje všechny klauzule, které se můžeme 'naučit' rezolucí z dané formule. Jde spíše o užitečný teoretický pohled na rezoluci, v aplikacích by bylo nepraktické konstruovat celý rezoluční uzávěr

Definice 5.2.9 (Rezoluční uzávěr)**.** *Rezoluční uzávěr* $\mathcal{R}(S)$ formule S je definován induktivně jako nejmenší množina klauzulí splňující:

- $C \in \mathcal{R}(S)$ pro všechna $C \in S$,
- jsou-li $C_1, C_2 \in \mathcal{R}(S)$ a je-li C rezolventa C_1, C_2, potom také $C \in \mathcal{R}(S)$.

Příklad 5.2.10. Spočtěme rezoluční uzávěr formule $S = \{\{p, \neg q, r\}, \{p, \neg r\}, \{\neg p, r\}, \{\neg p, \neg r\}, \{q, r\}\}$. Klauzule z S jsou modře, další klauzule získáváme postupným rezolvováním (první s první, druhá s první, druhá s druhou atd., přes všechny možné literály):

$$\begin{aligned}\mathcal{R}(S) = \{&\{p, \neg q, r\}, \{p, \neg r\}, \{\neg p, r\}, \{p, s\}, \{q, r\},\\ &\{p, \neg q\}, \{\neg q, r\}, \{r, \neg r\}, \{p, \neg p\}, \{r, s\}, \{p, r\}, \{p, q\}, \{r\}, \{p\}\}\end{aligned}$$

5.3 Korektnost a úplnost rezoluční metody

Rezoluční metoda je také korektní i úplná.

5.3.1 Korektnost rezoluce

Korektnost dokážeme snadno indukcí podle délky rezolučního důkazu.

Věta 5.3.1 (O korektnosti rezoluce)**.** *Je-li formule S rezolucí zamítnutelná, potom je S nesplnitelná.*

Důkaz. Nechť $S \vdash_R \square$ a vezměme nějaký rezoluční důkaz $C_0, C_1, \ldots, C_n = \square$. Předpokládejme pro spor, že S je splnitelná, tedy $\mathcal{V} \models S$ pro nějaké ohodnocení $\mathcal{V}$. Indukcí podle i dokážeme, že $\mathcal{V} \models C_i$. Pro $i = 0$ to platí, neboť $C_0 \in S$. Pro $i > 0$ máme dva případy:

- $C_i \in S$, v tom případě $\mathcal{V} \models C_i$ plyne z předpokladu, že $\mathcal{V} \models S$,
- C_i je rezolventou C_j, C_k, kde $j, k < i$: z indukčního předpokladu víme $\mathcal{V} \models C_j$ a $\mathcal{V} \models C_k$, $\mathcal{V} \models C_i$ plyne z korektnosti rezolučního pravidla.

(Alternativně bychom mohli v důkazu postupovat indukcí podle hloubky rezolučního stromu.) □

5.3.2 Strom dosazení

V důkazu úplnosti budeme potřebovat zkonstruovat rezoluční strom, jeho konstrukce je založena na tzv. *stromu dosazení*. *Dosazením* literálu do formule myslíme zjednodušení formule za předpokladu, že daný literál platí. Dosazení jsme už potkali v Sekci 3.3 při *jednotkové propagaci*: odstraníme klauzule obsahující tento literál, a z ostatních klauzulí odstraníme literál opačný.

Definice 5.3.2 (Dosazení literálu)**.** Je-li S formule a ℓ literál, potom *dosazením* ℓ do S myslíme formuli:

$$S^\ell = \{C \setminus \{\bar{\ell}\} \mid \ell \notin C \in S\}$$

Pozorování 5.3.3. *Zde shrneme několik jednoduchých faktů o dosazení:*

- S^ℓ *je výsledkem* jednotkové propagace *aplikované na* $S \cup \{\{\ell\}\}$.
- S^ℓ *neobsahuje v žádné klauzuli literál* ℓ *ani* $\bar{\ell}$ *(vůbec tedy neobsahuje prvovýrok z* ℓ*)*
- *Pokud* S *neobsahovala literál* ℓ *ani* $\bar{\ell}$*, potom* $S^\ell = S$.
- *Pokud* S *obsahovala jednotkovou klauzuli* $\{\bar{\ell}\}$*, potom* $\square \in S^\ell$*, tedy* S^ℓ *je sporná.*

Klíčovou vlastnost dosazení vyjadřuje následující lemma:

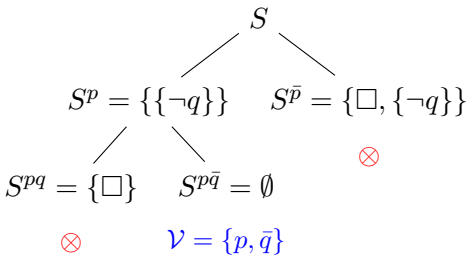
Lemma 5.3.4. *S je splnitelná, právě když je splnitelná S^ℓ nebo $S^{\bar{\ell}}$.*

Důkaz. Mějme ohodnocení $\mathcal{V} \models S$, to nemůže obsahovat ℓ i $\bar{\ell}$ (musí být konzistentní); bez újmy na obecnosti předpokládejme, že $\bar{\ell} \notin \mathcal{V}$, a ukažme, že $\mathcal{V} \models S^\ell$. Vezměme libovolnou klauzuli v S^ℓ. Ta je tvaru $C \setminus \{\bar{\ell}\}$ pro klauzuli $C \in S$ (neobsahující literál ℓ). Víme, že $\mathcal{V} \models C$, protože ale $\mathcal{V}$ neobsahuje $\bar{\ell}$, muselo ohodnocení $\mathcal{V}$ splnit nějaký jiný literál C, takže platí i $\mathcal{V} \models C \setminus \{\bar{\ell}\}$.

Naopak, předpokládejme že existuje ohodnocení $\mathcal{V}$ splňující S^ℓ (opět bez újmy na obecnosti). Protože se $\bar{\ell}$ (ani ℓ) nevyskytuje v S^ℓ, platí také $\mathcal{V} \setminus \{\bar{\ell}\} \models S^\ell$. Ohodnocení $\mathcal{V}' = (\mathcal{V} \setminus \{\bar{\ell}\}) \cup \{\ell\}$ potom splňuje každou klauzuli $C \in S$: pokud $\ell \in C$, potom $\ell \in C \cap \mathcal{V}'$ a $C \cap \mathcal{V}' \neq \emptyset$, jinak $C \cap \mathcal{V}' = (C \setminus \{\bar{\ell}\}) \cap \mathcal{V}' \neq \emptyset$ neboť $\mathcal{V} \setminus \{\bar{\ell}\} \models C \setminus \{\bar{\ell}\} \in S^\ell$. Ověřili jsme, že $\mathcal{V}' \models S$, tedy S je splnitelná. □

Zda je daná *konečná* formule splnitelná bychom tedy mohli zjišťovat rekurzivně (metodou *rozděl a panuj*), dosazením obou možných literálů pro (nějakou, třeba první) výrokovou proměnnou vyskytující se ve formuli, a rozvětvením výpočtu. V zásadě jde o podobný princip jako v algoritmu DPLL (viz Sekce 3.4). Výslednému stromu říkáme *strom dosazení*.

Příklad 5.3.5. Ilustrujme si tento koncept na příkladě, zkonstruujme strom dosazení pro formuli $S = \{\{p\}, \{\neg q\}, \{\neg p, \neg q\}\}$:



Jakmile větev obsahuje prázdnou klauzuli $\square$, je nesplnitelná a nemusíme v ní pokračovat. V listech jsou buď nesplnitelné teorie, nebo prázdná teorie: v tom případě z posloupnosti dosazení získáme splňující ohodnocení.

Z konstrukce je vidět, jak postupovat v případě konečné formule. Strom dosazení ale dává smysl, a následující důsledek platí, i pro nekonečné formule:

Důsledek 5.3.6. *Formule S (nad spočetným jazykem) je nesplnitelná, právě když každá větev stromu dosazení obsahuje prázdnou klauzuli $\square$.*

Důkaz. Pro konečnou formuli S plyne z diskuze výše, můžeme snadno dokázat indukcí podle velikosti $\mathrm{Var}(S)$:

- Je-li $|\mathrm{Var}(S)| = 0$, máme $S = \emptyset$ nebo $S = \{\square\}$, v obou případech je strom dosazení jednoprvkový a tvrzení platí.
- V indukčním kroku vybereme libovolný literál $\ell \in \mathrm{Var}(S)$ a aplikujeme Lemma 5.3.4.

Je-li S nekonečná a splnitelná, potom má splňující ohodnocení, to se 'shoduje' s odpovídající (nekonečnou) větví ve stromu dosazení. Je-li nekonečná a nesplnitelná, potom podle Věty o kompaktnosti existuje konečná část $S' \subseteq S$, která je také nesplnitelná. Po dosazení pro všechny proměnné z $\mathrm{Var}(S')$ bude v každé větvi $\square$, to nastane po konečně mnoha krocích. $\square$

5.3.3 Úplnost rezoluce

Věta 5.3.7 (O úplnosti rezoluce)**.** *Je-li S nesplnitelná, je rezolucí zamítnutelná (tj. $S \vdash_R \square$).*

Důkaz. Je-li S nekonečná, má z Věty o kompaktnosti konečnou nesplnitelnou část S'. Rezoluční zamítnutí S' je také rezolučním zamítnutím S. Předpokládejme tedy, že S je konečná.

Důkaz provedeme indukcí podle počtu proměnných v S. Je-li $|\mathrm{Var}(S)| = 0$, jediná možná nesplnitelná formule bez proměnných je $S = \{\square\}$ a máme jednokrokový důkaz $S \vdash_R \square$. Jinak vyberme $p \in \mathrm{Var}(S)$. Podle Lemmatu 5.3.4 jsou S^p i $S^{\bar{p}}$ nesplnitelné. Mají o jednu proměnnou méně, tedy podle indukčního předpokladu existují rezoluční stromy T pro $S^p \vdash_R \square$ a T' pro $S^{\bar{p}} \vdash_R \square$.

Ukážeme, jak ze stromu T vyrobit rezoluční strom $\widehat{T}$ pro $S \vdash_R \neg p$. Analogicky vyrobíme $\widehat{T'}$ pro $S \vdash_R p$ a potom už snadno vyrobíme rezoluční strom pro $S \vdash_R \square$: ke kořeni $\square$ připojíme kořeny stromů $\widehat{T}$ a $\widehat{T'}$ jako levého a pravého syna (tj. v posledním kroku rezolučního důkazu získáme $\square$ rezolucí z $\{\neg p\}$ a $\{p\}$).

Zbývá ukázat konstrukci stromu $\widehat{T}$: množina vrcholů i uspořádání jsou stejné, změníme jen některé klauzule ve vrcholech, a to přidáním literálu $\neg p$. Na každém listu stromu T je nějaká klauzule $C \in S^p$, a buď je $C \in S$, nebo není, ale $C \cup \{\neg p\} \in S$. V prvním případě necháme label stejný. Ve druhém případě přidáme do C *a do všech klauzulí nad tímto listem* literál $\neg p$. V listech jsou nyní jen klauzule z S, v kořeni jsme $\square$ změnili na $\neg p$. A každý vnitřní vrchol je nadále rezolventou svých synů. $\square$

Cvičení 5.2. Důkaz Věty o úplnosti rezoluce dává návod, jak rekurzivně 'vypěstovat' rezoluční zamítnutí. Rozmyslete si jak a proveďte na nějakém příkladě nesplnitelné formule.

5.4 LI-rezoluce a Horn-SAT

Začneme jiným pohledem na rezoluční důkaz, tzv. *lineárním důkazem.*

5.4.1 Lineární důkaz

Rezoluční důkaz můžeme kromě rezolučního stromu zorganizovat také ve formě tzv. *lineárního důkazu*, kde v každém kroku máme jednu *centrální* klauzuli, kterou rezolvujeme s *boční* ('side')

klauzulí, která je buď jednou z předchozích centrálních klauzulí, nebo axiomem z S. Rezolventa je potom novou centrální klauzulí.[4]

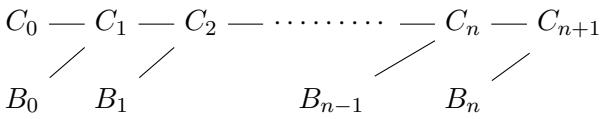
Definice 5.4.1 (Lineární důkaz)**.** *Lineární důkaz* (rezolucí) klauzule C z formule S je konečná posloupnost

$$\begin{bmatrix} C_0 \\ B_0 \end{bmatrix}, \begin{bmatrix} C_1 \\ B_1 \end{bmatrix}, \ldots, \begin{bmatrix} C_n \\ B_n \end{bmatrix}, C_{n+1}$$

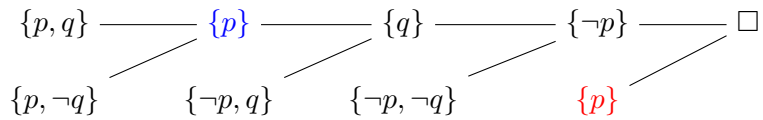
kde C_i říkáme *centrální* klauzule, C_0 je *počáteční*, $C_{n+1} = C$ je *koncová*, B_i jsou *boční* klauzule, a platí:

- $C_0 \in S$, pro $i \leq n$ je C_{i+1} rezolventou C_i a B_i,
- $B_0 \in S$, pro $i \leq n$ je $B_i \in S$ nebo $B_i = C_j$ pro nějaké $j < i$.

Lineární zamítnutí S je lineární důkaz $\square$ z S. Lineární důkaz můžeme znázornit takto:



Příklad 5.4.2. Zkonstruujme lineární zamítnutí formule $S = \{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}$ (tj. lineární důkaz $\square$ z S). Lineární důkaz může vypadat třeba takto:



Poslední boční klauzule $\{p\}$ (červeně) není z S, ale je rovna předchozí centrální klauzuli (modře).

Cvičení 5.3. Převeďte lineární důkaz z Příkladu 5.4.2 na rezoluční strom.

Poznámka 5.4.3. C má lineární důkaz z S, právě když $S \vdash_R C$.

Z lineárního důkazu snadno vyrobíme rezoluční strom. Indukcí podle délky důkazu: základ indukce je zřejmý, a máme-li boční klauzuli B_i která není axiomem z S, potom $B_i = C_j$ pro nějaké $j < i$ a stačí připojit místo B_i rezoluční strom pro důkaz C_j z S. Všimněte si, že z toho plyne i *korektnost* lineární rezoluce.

Důkaz opačné implikace neuvedeme. Plyne z *úplnosti* lineární rezoluce, jejíž důkaz najdete v učebnici *A. Nerode, R. Shore: Logic for Applications* [3].

5.4.2 LI-rezoluce

V obecném lineárním důkazu může být každá následující boční klauzule buď axiom z S nebo jedna z předchozích centrálních klauzulí. Pokud zakážeme druhou možnost, budeme-li tedy požadovat, aby všechny boční klauzule byly z S, dostaneme tzv. *LI (linear-input)* rezoluci:

[4] Zatímco konstrukci rezolučního stromu lze snadno popsat rekurzivně, lineární důkaz lépe odpovídá procedurálnímu výpočtu. Jde jen o to, jak najít vhodnou boční klauzuli.

Definice 5.4.4 (LI-důkaz)**.** *LI-důkaz* (rezolucí) klauzule C z formule S je lineární důkaz

$$\begin{bmatrix} C_0 \\ B_0 \end{bmatrix}, \begin{bmatrix} C_1 \\ B_1 \end{bmatrix}, \ldots, \begin{bmatrix} C_n \\ B_n \end{bmatrix}, C$$

ve kterém je každá boční klauzule B_i axiom z S. Pokud LI-důkaz existuje, říkáme, že je C *LI-dokazatelná* z S, a píšeme $S \vdash_{LI} C$. Pokud $S \vdash_{LI} \square$, je S *LI-zamítnutelná*.

Poznámka 5.4.5. LI-důkaz přímo dává rezoluční strom (všechny listy jsou axiomy), a to ve speciálním tvaru, kterému bychom mohli říkat 'chlupatá cesta'. A naopak, z rezolučního stromu ve tvaru chlupaté cesty okamžitě získáme LI-důkaz: vrcholy na cestě jsou centrální klauzule, chlupy jsou boční klauzule.

Zatímco *lineární rezoluce*[5] je jen jiný pohled na obecný rezoluční důkaz, *LI-rezoluce* přináší zásadní omezení: ztrácíme *úplnost* (ne každá nesplnitelná formule má LI-zamítnutí). Na druhou stranu, LI-důkazy je jednodušší konstruovat.[6]

5.4.3 Úplnost LI-rezoluce pro Hornovy formule

Jak si nyní ukážeme, LI-rezoluce je *úplná pro Hornovy formule*. A jak uvidíme v následující sekci, je základem interpreterů jazyka Prolog, který s Hornovými formulemi pracuje. Nejprve připomeňme terminologii týkající se hornovskosti a také programů, a to v množinové reprezentaci:

- *Hornova klauzule* je klauzule obsahující nejvýše jeden pozitivní literál.
- *Hornova formule* je (konečná, nebo i nekonečná) množina Hornových klauzulí.
- *Fakt* je pozitivní jednotková (Hornova) klauzule, tj. $\{p\}$, kde p je výroková proměnná.
- *Pravidlo* je (Hornova) klauzule s právě jedním pozitivním a alespoň jedním negativním literálem.
- Pravidlům a faktům říkáme *programové klauzule*.
- *Cíl* je neprázdná (Hornova) klauzule bez pozitivního literálu.[7]

Bude se nám hodit následující jednoduché pozorování:

Pozorování 5.4.6. *Je-li Hornova formule S nesplnitelná a $\square \notin S$, potom obsahuje fakt i cíl.*

Důkaz. Neobsahuje-li fakt, můžeme ohodnotit všechny proměnné 0; neobsahuje-li cíl, ohodnotíme 1. □

Nyní vyslovíme a dokážeme Větu o úplnosti LI-rezoluce pro Hornovské formule. Důkaz dává také návod, jak LI-zamítnutí zkonstruovat, a to na základě průběhu jednotkové propagace. Tento postup ilustrujeme na příkladu níže, který můžete sledovat souběžně s čtením důkazu.

[5] Tj. dokazovací systém založený na hledání *lineárních* důkazů resp. zamítnutí.
[6] V každém kroku máme k volbě jen klauzule z S, nikoliv předchozí dokázané centrální klauzule.
[7] Připomeňme, že dokazujeme *sporem*, tedy *cíl* je negací toho, co bychom chtěli dokázat.

Věta 5.4.7 (O úplnosti LI-rezoluce pro Hornovy formule)**.** *Je-li Hornova formule T splnitelná, a $T \cup \{G\}$ je nesplnitelná pro cíl G, potom $T \cup \{G\} \vdash_{LI} \square$, a to LI-zamítnutím, které začíná cílem G.*

Důkaz. Podobně jako ve Větě o úplnosti rezoluce můžeme díky Větě o kompaktnosti předpokládat, že T je konečná. Důkaz (konstrukci LI-zamítnutí) provedeme indukcí podle počtu proměnných v T.

Z Pozorování 5.4.6 plyne, že T obsahuje fakt $\{p\}$ pro nějakou výrokovou proměnnou p. Protože $T \cup \{G\}$ je nesplnitelná, je podle Lemmatu 5.3.4 nesplnitelná také $(T \cup \{G\})^p = T^p \cup \{G^p\}$, kde $G^p = G \setminus \{\neg p\}$.

Pokud $G^p = \square$, potom $G = \{\neg p\}$, $\square$ je rezolventa G a $\{p\} \in T$, a máme jednokrokové LI-zamítnutí $T \cup \{G\}$ (to je báze indukce).

Jinak využijeme indukčního předpokladu. Všimněte si, že formule T^p je splnitelná (stejným ohodnocením jako T, neboť to musí obsahovat p kvůli faktu $\{p\}$, tedy neobsahuje $\neg p$) a má méně proměnných než T. Tedy podle indukčního předpokladu existuje LI-odvození $\square$ z $T^p \cup \{G^p\}$ začínající $G^p = G \setminus \{\neg p\}$.

Hledané LI-zamítnutí $T \cup \{G\}$ začínající G zkonstruujeme (podobně jako v důkazu Věty o úplnosti rezoluce) přidáním literálu $\neg p$ do všech listů, které už nejsou v $T \cup \{G\}$ (tedy vznikly odebráním $\neg p$), a do všech vrcholů nad nimi. Tím získáme $T \cup \{G\} \vdash_{LI} \neg p$, na závěr přidáme boční klauzuli $\{p\}$ a odvodíme $\square$. $\square$

Příklad 5.4.8. Mějme (splnitelnou, hornovskou) teorii T, kterou zapíšeme v množinové reprezentaci jako formuli $T = \{\{p, \neg r, \neg s\}, \{\neg q, r\}, \{q, \neg s\}, \{s\}\}$. Představte si, že chceme dokázat, že v teorii T platí $p \wedge q$.[8] V rezoluční metodě uvážíme cíl $G = \{\neg p, \neg q\}$ a ukážeme, že $T \cup \{G\} \vdash_{LI} \square$.

Dle návodu z důkazu najdeme ve formuli T fakt, a provedeme pomocí něho jednotkovou propagaci v T i v cíli G. Postup opakujeme, dokud není formule prázdná:

- $T = \{\{p, \neg r, \neg s\}, \{\neg q, r\}, \{q, \neg s\}, \{s\}\}$, $G = \{\neg p, \neg q\}$
- $T^s = \{\{p, \neg r\}, \{\neg q, r\}, \{q\}\}$, $G^s = \{\neg p, \neg q\}$
- $T^{sq} = \{\{p, \neg r\}, \{r\}\}$, $G^{sq} = \{\neg p\}$
- $T^{sqr} = \{\{p\}\}$, $G^{sqr} = \{\neg p\}$
- $T^{sqrp} = \emptyset$, $G^{sqrp} = \square$

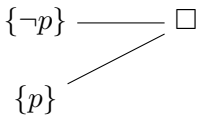
Nyní zpětným postupem sestrojíme rezoluční zamítnutí:

- $T^{sqrp}, G^{sqrp} \vdash_{LI} \square$:

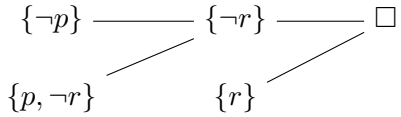
$$\square$$

- $T^{sqr}, G^{sqr} \vdash_{LI} \square$:

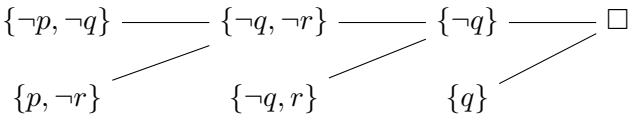
[8] Tj. v Prologu bychom položili ‘dotaz’ (‘query’): `?-p,q.`



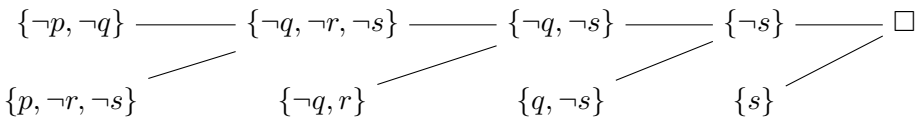
- $T^{sq}, G^{sq} \vdash_{LI} \square$:



- $T^{s}, G^{s} \vdash_{LI} \square$:



- $T, G \vdash_{LI} \square$



5.4.4 Program v Prologu

Ačkoliv skutečná síla Prologu vychází z tzv. *unifikace* a z rezoluce v predikátové logice, ukážeme si jak Prolog využívá rezoluční metodu na příkladě *výrokového* programu. Adaptace na predikáty bude později přímočará.

Program v Prologu je Hornova formule obsahující pouze *programové klauzule*, tj. *fakta* nebo *pravidla*. *Dotaz* je konjunkce faktů, negace dotazu je *cíl*.

Příklad 5.4.9. Jako příklad programu v Prologu využijeme teorii (formuli) T a dotaz $p \wedge q$ z Příkladu 5.4.8. Například klauzuli $\{p, \neg r, \neg s\}$, která je ekvivalentní $r \wedge s \to p$, zapíšeme v Prologu jako: `p:-r,s.`

```
p:-r,s.
r:-q.
q:-s.
s.
```

A programu položíme dotaz:

```
?-p,q.
```

Důsledek 5.4.10. *Mějme program P a dotaz $Q = p_1 \wedge \cdots \wedge p_n$, a označme $G = \{\neg p_1, \ldots, \neg p_n\}$ (tj. $G \sim \neg Q$). Následující podmínky jsou ekvivalentní:*

- $P \models Q$,
- $P \cup \{G\}$ *je nesplnitelná,*
- $P \cup \{G\} \vdash_{LI} \square$*, a existuje LI-zamítnutí začínající cílem* G*.*

Důkaz. Ekvivalence prvních dvou podmínek je důkaz sporem, ekvivalence druhé a třetí je Věta o úplnosti LI-rezoluce pro Hornovy formule. (Všimněte si, že Program je vždy splnitelný.) $\square$

Část II

Predikátová logika

Kapitola 6

Syntaxe a sémantika predikátové logiky

Kurzy logiky vesměs začínají výrokovou logikou, která je pro svou jednoduchost vhodnější k prvnímu seznámení. Plná síla logiky v informatice se ale projeví teprve s použitím logiky predikátové. Začněme neformálním úvodem, ve kterém ilustrujeme základní aspekty predikátové logiky. K formálnímu výkladu se vrátíme v následujících sekcích.

6.1 Úvod

Připomeňme, že ve výrokové logice jsme popisovali svět pomocí *výroků* složených z *prvovýroků*—odpovědí na zjišťovací (ano/ne) otázky o světě. V predikátové logice (prvního řádu)[1] jsou základním stavebním kamenem *proměnné* reprezentující *individua*—nedělitelné objekty z nějaké množiny: např. přirozená čísla, vrcholy grafu, nebo stavy mikroprocesoru.

Tato individua mohou mít určité vlastnosti a vzájemné vztahy, kterým říkáme *predikáty*, např. 'Leaf(x)' nebo '$Edge(x, y)$' mluvíme-li o grafu, nebo '$x \leq y$' v přirozených číslech. Kromě toho mohou individua vstupovat do funkcí, např. 'lowest_common_ancestor(x, y)' v zakořeněném stromu, 'succ(x)' nebo '$x + y$' v přirozených číslech, a být *konstantami* se speciálním významem, např. 'root' v zakořeněném stromu, '0' v přirozených číslech.

Atomické formule popisují predikát (včetně predikátu *rovnosti* $=$) o proměnných nebo o *termech* ('výrazech' složených[2] z funkcí popř. konstant). A složitější tvrzení (*formule*) budujeme z atomických formulí pomocí logických spojek, a dvou *kvantifikátorů*:

- $\forall x$ "pro všechna individua (reprezentovaná proměnnou x)," a
- $\exists x$ "existuje individuum (reprezentované proměnnou x)".

Uveďme příklad: tvrzení *"Každý, kdo má dítě, je rodič."* bychom mohli formalizovat následující formulí:

$$(\forall x)((\exists y)\text{child\_of}(y, x) \rightarrow \text{is\_parent}(x))$$

[1] V logice druhého řádu máme také proměnné reprezentující množiny individuí nebo i množiny n-tic, tj. relace na množině individuí.

[2] Podobně jako vytváříme aritmetické výrazy.

kde child_of(y, x) je binární predikát vyjadřující, že individuum reprezentované proměnnou y je dítětem individua reprezentovaného proměnnou x, a is_parent(x) je unární predikát (tj. 'vlastnost') vyjadřující, že individuum reprezentované x je rodič.

Jak je to s platností této formule? To záleží na konkrétním *modelu* světa/systému, který nás zajímá. Model je (neprázdná) množina objektů spolu s unární relací (tj. podmnožinou) *interpretující unární relační symbol* is_parent a binární relací interpretující *binární relační symbol* child_of. Tyto relace mohou být obecně jakékoliv a snadno sestrojíme model, kde formule neplatí.[3] Pokud ale modelujeme například všechny lidi na světě, a relace mají svůj přirozený význam, potom formule bude platit.[4]

Podívejme se na ještě jeden příklad, tentokrát i s funkčními symboly a s konstantním symbolem: "Je-li $x_1 \leq y_1$ a $x_2 \leq y_2$, potom platí $(y_1 \cdot y_2) - (x_1 \cdot x_2) \geq 0$." Výsledná formule by mohla vypadat takto:

$$\varphi = (x_1 \leq y_1) \wedge (x_2 \leq y_2) \to ((y_1 \cdot y_2) + (-(x_1 \cdot x_2)) \geq 0)$$

Vidíme dva binární relační symboly $(\leq, \geq)$, binární funkční symbol $+$, unární funkční symbol $-$, a konstantní symbol 0.

Modelem, ve kterém formule platí, je množina přirozených čísel $\mathbb{N}$ s binárními relacemi $\leq^{\mathbb{N}}, \geq^{\mathbb{N}}$, binárními funkcemi $+^{\mathbb{N}}, \cdot^{\mathbb{N}}$, unární funkcí $-^{\mathbb{N}}$, a konstantou $0^{\mathbb{N}} = 0$. Vezmeme-li ale podobně množinu celých čísel, formule už platit nebude.

Poznámka 6.1.1. Mohli bychom chápat symbol $-$ jako binární operaci, obvykle se ale zavádí jako unární. Pro konstantní symbol 0 používáme (jak je zvykem) stejný symbol, jako pro přirozené číslo 0. Ale pozor, v našem modelu může být tento konstantní symbol interpretován jako jiné číslo, nebo náš model vůbec nemusí sestávat z čísel!

Ve formuli nejsou žádné kvantifikátory (takovým formulím říkáme *otevřené*), proměnné x_1, x_2, y_1, y_2 jsou *volné proměnné* této formule (nejsou *vázané* žádným kvantifikátorem), píšeme $\varphi(x_1, x_2, y_1, y_2)$. Sémantiku této formule chápeme stejně, jako formule

$$(\forall x_1)(\forall x_2)(\forall y_1)(\forall y_2)\varphi(x_1, x_2, y_1, y_2)$$

Výraz $(y_1 \cdot y_2) + (-(x_1 \cdot x_2))$ je příkladem *termu*, výrazy $(x_1 \leq y_1)$, $(x_2 \leq y_2)$ a $((y_1 \cdot y_2) + (-(x_1 \cdot x_2)) \geq 0)$ jsou *atomické (pod)formule*. V čem spočívá rozdíl? Máme-li konkrétní model, a konkrétní *ohodnocení proměnných* individui (prvky) tohoto modelu, potom atomickým formulím lze přiřadit pravdivostní hodnotu. Lze je tedy kombinovat s logickými spojkami do složitějších 'logických výrazů', tj. formulí. Na druhou stranu 'výsledkem' termu (při daném ohodnocení) je nějaké konkrétní individuum z modelu.

Upozorníme ještě na to, že v zápisu formule φ jsme použili infixový zápis pro funkční symboly $+, \cdot$ a pro relace $\leq, \geq$, a podobné konvence o uzávorkování jako ve výrokové logice. Jinak bychom formuli φ zapsali takto:

$$((\leq (x_1, y_1) \wedge \leq (x_2, y_2)) \to \leq (+(\cdot(y_1, y_2), -(\cdot(x_1, x_2))), 0))$$

Cvičení 6.1. Najděte vhodnou definici pojmu *stromu formule* (zobecňující *strom výroku* z výrokové logiky) a nakreslete strom formule $(\forall x_1)(\forall x_2)(\forall y_1)(\forall y_2)\varphi(x_1, x_2, y_1, y_2)$.

[3] Vezměme například jednoprvkovou množinu $A = \{a\}$, a relace child_of$^A = \{a\}$, parent$^A = \emptyset$, tedy jediný objekt je svým vlastním dítětem, ale není rodičem.

[4] Při formalizaci musíme být velmi opatrní, abychom nepřidali dodatečné předpoklady, které v modelovaném systému nemusí platit. Zde se například schovává předpoklad, že má-li někdo dítě, musí to být jeho dítě.

Nyní začneme tím, že formalizujeme tento koncept "*modelu*", tzv. *strukturu*. Zbytek kapitoly sleduje osnovu výkladu o výrokové logice: představíme syntaxi, následně sémantiku, a nakonec pokročilejší vlastnosti formulí, teorií, a struktur. Na závěr si ukážeme jednu jednoduchou, ale velmi užitečnou aplikaci predikátové logiky, takzvanou *definovatelnost* podmnožin a relací, která je základem *relačních databází* (např. SQL), a ještě jednou se podíváme na vztah výrokové a predikátové logiky.

6.2 Struktury

Nejprve specifikujeme, jakého *typu* bude daná struktura, tj. jaké bude mít relace, funkce (jakých arit) a konstanty, a jaké symboly pro ně budeme používat. Této formální specifikaci se někdy říká *typ*, my budeme říkat *signatura*.[5] Připomeňme, že *konstanty* můžeme chápat jako funkce arity 0 (tj. funkce bez vstupů).

Definice 6.2.1. *Signatura* je dvojice $\langle \mathcal{R}, \mathcal{F} \rangle$, kde $\mathcal{R}, \mathcal{F}$ jsou disjunktní množiny symbolů (*relační* a *funkční*, ty zahrnují *konstantní*) spolu s danými aritami (tj. danými funkcí $\mathrm{ar}\colon \mathcal{R} \cup \mathcal{F} \to \mathbb{N}$) a neobsahující symbol '=' (ten je rezervovaný pro *rovnost*).

Často ale budeme signaturu zapisovat jen výčtem symbolů, bude-li jejich arita a to, zda jsou relační nebo funkční, zřejmé z kontextu. Uveďme několik příkladů signatur:

- $\langle E \rangle$ signatura *grafů*: E je binární relační symbol (struktury jsou uspořádané grafy),
- $\langle \leq \rangle$ signatura *částečných uspořádání*: stejná jako signatura grafů, jen jiný symbol,[6]
- $\langle +, -, 0 \rangle$ signatura *grup*: $+$ je binární funkční, $-$ unární funkční, 0 konstantní symbol
- $\langle +, -, 0, \cdot, 1 \rangle$ signatura *těles*: $\cdot$ je binární funkční, 1 konstantní symbol
- $\langle +, -, 0, \cdot, 1, \leq \rangle$ signatura *uspořádaných těles*: $\leq$ je binární relační symbol,
- $\langle -, \wedge, \vee, \bot, \top \rangle$ signatura *Booleových algeber*: $\wedge, \vee$ jsou binární funkční symboly, $\bot, \top$ jsou konstantní symboly,
- $\langle S, +, \cdot, 0, \leq \rangle$ signatura *aritmetiky*: S je unární funkční symbol ('successor').

Kromě běžných symbolů relací, funkcí a konstant (známých např. z logiky nebo z aritmetiky) typicky používáme pro relační symboly $P, Q, R, \ldots$, pro funkční symboly $f, g, h, \ldots$, a pro konstantní symboly $c, d, a, b, \ldots$

Strukturu dané signatury získáme tak, že na nějaké neprázdné *doméně* zvolíme *realizace* (také říkáme *interpretace*) všech relačních a funkčních symbolů (a konstant), tj. konkrétní relace resp. funkce příslušných arit. (V případě konstantního symbolu je jeho realizací zvolený prvek z domény.)[7]

[5] Signaturu si můžete představovat analogicky definici *třídy* v OOP, struktury potom odpovídají *objektům* této třídy (v 'programovacím jazyce' teorie množin).

[6] Ne každá struktura v této signatuře je částečné uspořádání, k tomu ještě potřebujeme, aby splňovala příslušné *axiomy*.

[7] Na tom, jaké konkrétní symboly v signatuře použijeme, nezáleží, můžeme je interpretovat libovolně. Například to, že máme symbol $+$ neznamená, že by jeho interpretace musela mít cokoliv společného se sčítáním (tedy kromě toho, že to bude také binární funkce).

Příklad 6.2.2. Formální definice *struktury* je uvedena níže, nejprve si ukážeme několik příkladů:

- Struktura v prázdné signatuře $\langle\ \rangle$ je libovolná neprázdná množina.[8] (Nemusí být konečná, dokonce ani spočetná!)
- Struktura v signatuře grafů je $\mathcal{G} = \langle V, E\rangle$, kde $V \neq \emptyset$ a $E \subseteq V^2$, říkáme jí *orientovaný graf*.
  - Je-li E ireflexivní a symetrická, jde o *jednoduchý* graf (tj. neorientovaný, bez smyček).
  - Je-li E reflexivní, tranzitivní, a antisymetrická, jde o *částečné uspořádání*.
  - Je-li E reflexivní, tranzitivní, a symetrická, mluvíme o *ekvivalenci*.
- Struktury v signatuře částečných uspořádání jsou tytéž, jako v signatuře grafů, signatury se liší jen použitým symbolem. (Tedy ne každá struktura v signatuře částečných uspořádání je částečné uspořádání!)
- Struktury v signatuře grup jsou například následující *grupy*:
  - $\underline{\mathbb{Z}_n} = \langle \mathbb{Z}_n, +, -, 0\rangle$, *aditivní grupa celých čísel modulo n* (operace jsou modulo n).[9]
  - $\mathcal{S}_n = \langle \mathrm{Sym}_n, \circ, ^{-1}, \mathrm{id}\rangle$ je *symetrická grupa* (grupa všech permutací) na n prvcích.
  - $\underline{\mathbb{Q}^*} = \langle \mathbb{Q} \setminus \{0\}, \cdot, ^{-1}, 1\rangle$ je *multiplikativní grupa (nenulových) racionálních čísel*. Všimněte si, že interpretací *symbolu* 0 je *číslo* 1.

  Všechny tyto struktury *splňují axiomy teorie grup*, snadno ale najdeme jiné struktury, které tyto axiomy nesplňují, a nejsou tedy grupami. Například změníme-li ve struktuře $\mathbb{Z}_n$ interpretaci symbolu $+$ na funkci $\cdot$ (modulo n).
- Struktury $\underline{\mathbb{Q}} = \langle \mathbb{Q}, +, -, 0, \cdot, 1, \leq\rangle$ a $\underline{\mathbb{Z}} = \langle \mathbb{Z}, +, -, 0, \cdot, 1, \leq\rangle$, se standardními operacemi a uspořádáním, jsou v signatuře uspořádaných těles (ale jen první z nich je uspořádaným tělesem).
- $\underline{\mathcal{P}(X)} = \langle \mathcal{P}(X), ^{\complement}, \cap, \cup, \emptyset, X\rangle$, tzv. *potenční algebra* nad množinou X, je to struktura v signatuře Booleových algeber. (*Booleova algebra* je to pokud $X \neq \emptyset$.)
- $\underline{\mathbb{N}} = \langle \mathbb{N}, S, +, \cdot, 0, \leq\rangle$, kde $S(x) = x + 1$, a ostatní symboly jsou interpretovány standardně, je *standardní model aritmetiky*.

Definice 6.2.3 (Struktura)**.** *Struktura v signatuře* $\langle \mathcal{R}, \mathcal{F}\rangle$ je trojice $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}}\rangle$, kde

- A je neprázdná množina, říkáme jí *doména* (také *univerzum*),
- $\mathcal{R}^{\mathcal{A}} = \{R^{\mathcal{A}} \mid R \in \mathcal{R}\}$ kde $R^{\mathcal{A}} \subseteq A^{\mathrm{ar}(R)}$ je *interpretace* relačního symbolu R,

[8] Jak uvidíme v definici níže, formálně vzato je to trojice $\langle A, \emptyset, \emptyset\rangle$, ale tento rozdíl budeme zanedbávat.

[9] Zde $\underline{\mathbb{Z}_n}$ znamená strukturu, zatímco $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ jen její doménu. Často se ale toto nerozlišuje a symbol $\mathbb{Z}_n$ se používá jak pro celou strukturu, tak jen její doménu. Podobně $+, -, 0$ jsou jak symboly, tak i jejich interpretace. To je běžně používané značení, je klíčové být si vždy vědomi toho, v jakém významu daný symbol na daném místě používáme.

- $\mathcal{F}^{\mathcal{A}} = \{f^{\mathcal{A}} \mid f \in \mathcal{F}\}$ kde $f^{\mathcal{A}}\colon A^{\mathrm{ar}(f)} \to A$ je *interpretace* funkčního symbolu f (speciálně pro konstantní symbol $c \in \mathcal{F}$ máme $c^{\mathcal{A}} \in A$).

Cvičení 6.2. Uvažme signaturu *n konstant* $\langle c_1, c_2, \ldots, c_n\rangle$. Jak vypadají struktury v této signatuře? Popište např. všechny nejvýše pětiprvkové struktury v signatuře tří konstant. (Interpretace konstant nemusí být různé!) A jak je tomu v případě signatury *spočetně mnoha konstant* $\langle c_1, c_2, \ldots\rangle = \langle c_i \mid i \in \mathbb{N}\rangle$?

6.3 Syntaxe

V této sekci představíme syntaxi predikátové logiky (prvního řádu). Srovnejte co má syntaxe společného, a jak se liší, od syntaxe výrokové logiky.

6.3.1 Jazyk

Při specifikaci jazyka nejprve stanovíme, v jakého typu jsou struktury, které chceme popisovat, tj. určíme *signaturu*. Dále přidáme informaci, zda je jazyk *s rovností* nebo ne, tj. zda ve formulích můžeme také používat symbol '=' vyjadřující rovnost (identitu) prvků v doméně struktur.[10] Do jazyka patří následující:

- spočetně mnoho *proměnných* $x_0, x_1, x_2, \ldots$ (ale píšeme také $x, y, z, \ldots$; množinu všech proměnných označíme Var),
- *relační*, *funkční* a *konstantní symboly* ze signatury, a symbol $=$ jde-li o jazyk s rovností,
- *univerzální* a *existenční kvantifikátory* $(\forall x), (\exists x)$ pro každou proměnnou $x \in \mathrm{Var}$,[11]
- symboly pro logické spojky $\neg, \wedge, \vee, \to, \leftrightarrow$ a závorky $(,)$.

Podobně jako symbol $\square$ zastupující libovolnou binární logickou spojku budeme někdy psát (Qx) pro kvantifikátor $(\forall x)$ nebo $(\exists x)$.

Symbolům ze signatury říkáme *mimologické*, ostatní jsou *logické*. Jazyk musí obsahovat alespoň jeden relační symbol (buď rovnost, nebo v signatuře).[12]

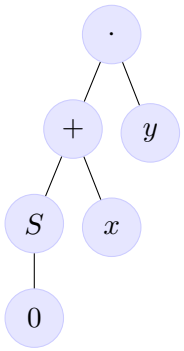
Jazyk tedy specifikujeme pomocí signatury a informace 's rovností' (popř. 'bez rovnosti'). Například:

- Jazyk $L = \langle\rangle$ s rovností je jazyk *čisté rovnosti*,
- jazyk $L = \langle c_0, c_1, c_2, \ldots\rangle$ s rovností je jazyk *spočetně mnoha konstant*,
- jazyk *uspořádání* je $\langle\leq\rangle$ s rovností,
- jazyk *teorie grafů* je $\langle E\rangle$ s rovností,
- jazyky *teorie grup, teorie těles, teorie uspořádaných těles, Booleových algeber, aritmetiky* jsou jazyky s rovností odpovídající signaturám z Příkladu 6.2.2

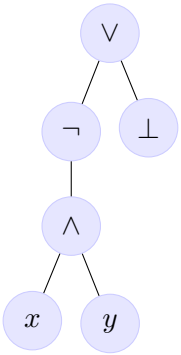
[10] Ve většině aplikací budeme používat jazyky s rovností. V některých speciálních oblastech se ale hodí rovnost nemít. Například pokud se zabýváme velmi rychlými výpočetními modely: zjistit, které proměnné se sobě rovnají, vyžaduje najít tranzitivní uzávěr rovností daných formulí, což je relativně výpočetně náročný problém.

[11] Kvantifikátor chápeme jako jediný symbol, tedy $(\forall x)$ *neobsahuje* proměnnou x. Někdy se také používají symboly $\forall_x, \exists_x$.

[12] Jinak bychom v jazyce nemohli vybudovat žádná 'tvrzení' (*formule*), viz níže.



(a) $(S(0)+x)\cdot y$ v jazyce aritmetiky



(b) $\neg(x\wedge y)\vee\bot$ v jazyce Booleových algeber

Obrázek 6.1: Strom termu

6.3.2 Termy

Termy jsou syntaktické 'výrazy' složené z proměnných, konstantních symbolů a funkčních symbolů.

Definice 6.3.1 (Termy)**.** *Termy* jazyka L jsou konečné nápisy definované induktivně:

- každá proměnná a každý konstantní symbol z L je term,
- je-li f funkční symbol z L arity n a jsou-li $t_1,\ldots,t_n$ termy, potom nápis $f(t_1,t_2,\ldots,t_n)$ je také term.

Množinu všech *termů* jazyka L označíme Term_L.

Při zápisu termů obsahujících binární funkční symbol můžeme používat *infixový* zápis, např. (t_1+t_2) znamená $+(t_1,t_2)$. Závorky někdy vynecháváme, je-li struktura termu ('priorita operátorů') zřejmá.

Podterm je podřetězec termu, který je sám termem (je to tedy buď celý term, nebo se vyskytl jako nějaké t_i při konstrukci termu).

Pokud term neobsahuje proměnnou, říkáme mu *konstantní (ground)*, například $((S(0)+S(0))\cdot S(S(0)))$ je konstantní term v jazyce aritmetiky.[13]

Strom termu t, označme $\mathrm{Tree}(t)$, je definován podobně jako strom výroku, v listech jsou proměnné nebo konstantní symboly, ve vnitřních vrcholech jsou funkční symboly, jejichž arita je rovna počtu synů.

Příklad 6.3.2. Nakresleme stromy termů (a) $(S(0)+x)\cdot y$ v jazyce aritmetiky, (b) $\neg(x\wedge y)\vee\bot$ v jazyce Booleových algeber. Zde $\neg,\wedge,\vee$ nejsou logické spojky z jazyka, ale mimologické symboly ze signatury Booleových algeber (byť používáme stejné symboly)! Termy v tomto jazyce můžeme chápat jako výrokové formule (s konstantami pro spor a tautologii), viz Sekce 6.9. Na obrázku 6.1 jsou nakresleny stromy těchto termů.

Není těžké uhádnout, jaká bude *sémantika* termů. Máme-li konkrétní strukturu, odpovídá term funkci na její doméně: vstupem je ohodnocení proměnných prvky domény, konstantní a funkční symboly jsou nahrazeny jejich interpretacemi, a výstupem je hodnota (prvek domény) v kořeni. Formálněji ale až v Sekci 6.4.

[13] Pozor, termy jsou čistě syntaktické, můžeme používat jen symboly z jazyka, nikoliv prvky struktury, tedy např. $(1+1)\cdot 2$ *není* term v jazyce aritmetiky! (Mohli bychom ale *definovat* nové konstantní symboly $1,2$ jako zkratky za $S(0)$ a $S(S(0))$ a *rozšířit* tak náš jazyk, viz Sekce 6.7.1.)

6.3.3 Formule

Termům nelze v žádném smyslu přiřadit pravdivostní hodnotu, k tomu potřebujeme *predikát* (relační symbol nebo rovnost), který mluví o 'vztahu' termů: v konkrétní struktuře při konkrétním ohodnocení proměnných prvky z domény je tento vztah buď splněn, nebo nesplněn.

Nejjednoduššími *formulemi* jsou *atomické formule*. Z nich potom vybudujeme pomocí logických spojek a kvantifikátorů všechny formule.

Definice 6.3.3 (Atomická formule)**.** *Atomická formule* jazyka L je nápis $R(t_1, \ldots, t_n)$, kde R je n-ární relační symbol z L (včetně $=$ jde-li o jazyk s rovností) a $t_i \in \mathrm{Term}_L$.

U binárních relačních symbolů často používáme infixový zápis, např. atomickou formuli $\leq(x, y)$ zapíšeme jako $x \leq y$, a (je-li jazyk s rovností) místo $=(t_1, t_2)$ budeme psát $t_1 = t_2$.

Příklad 6.3.4. Uveďme několik příkladů atomických formulí:

- $R(f(f(x)), c, f(d))$ kde R je ternární relační, f unární funkční, c, d konstantní symboly,
- $(x \cdot x) + (y \cdot y) \leq (x + y) \cdot (x + y)$ v jazyce uspořádaných těles,
- $x \cdot y \leq (S(0) + x) \cdot y$ v jazyce aritmetiky,
- $\neg(x \wedge y) \vee \bot = \bot$ v jazyce Booleových algeber

Definice 6.3.5 (Formule)**.** *Formule* jazyka L jsou konečné nápisy definované induktivně:

- každá atomická formule jazyka L je formule,
- je-li φ formule, potom $(\neg\varphi)$ je také formule,
- jsou-li φ, ψ formule, potom $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, a $(\varphi \leftrightarrow \psi)$ jsou také formule,
- je-li φ formule a x proměnná, potom $((\forall x)\varphi)$ a $((\exists x)\varphi)$ jsou také formule.

Podformule je podřetězec, který je sám o sobě formulí. *Strom formule*, označíme $\mathrm{Tree}(\varphi)$, je definován takto: strom atomické formule $\varphi = R(t_1, \ldots, t_n)$ má v kořeni relační symbol R, a k němu jsou připojeny stromy $\mathrm{Tree}(t_i)$. Není-li φ atomická, strom zkonstruujeme obdobně jako strom výroku.[14] Při zápisu formulí používáme obdobné konvence jako ve výrokové logice, přičemž kvantifikátory mají stejnou prioritu jako $\neg$ (vyšší než ostatní logické spojky). Místo $((\forall x)\varphi)$ tedy můžeme psát $(\forall x)\varphi$.[15]

Příklad 6.3.6. Příkladem formule v jazyce aritmetiky je $(\forall x)(x \cdot y \leq (S(0) + x) \cdot y)$. Její strom je znázorněn na Obrázku 6.2.

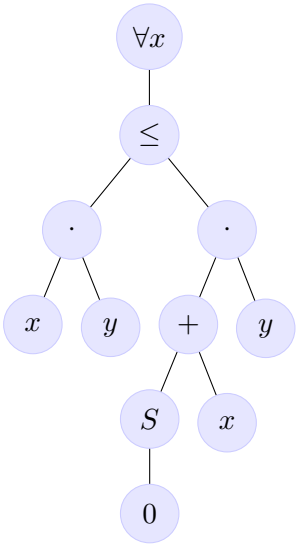
Volné a vázané proměnné

Význam formule[16] může, nebo nemusí záviset na proměnných, které se v ní vyskytují: srovnejte $x \leq 0$ a $(\exists x)(x \leq 0)$ (a co teprve $x \leq 0 \vee (\exists x)(x \leq 0)$). Nyní tento koncept upřesníme a zavedeme potřebnou terminologii.

[14] Kvantifikátory mají, podobně jako negace, jediného syna.

[15] Někdy se také nepíší závorky v kvantifikátorech, tj. jen $\forall x\varphi$, my je ale pro přehlednost psát budeme.

[16] Přesněji, její *pravdivostní hodnota*, kterou formálně definujeme níže v Sekci 6.4.3.



Obrázek 6.2: Strom formule $(\forall x)(x \cdot y \leq (S(0) + x) \cdot y)$

Výskytem proměnné x ve formuli φ myslíme list $\mathrm{Tree}(\varphi)$ označený x. [17] Výskyt je *vázaný*, je-li součástí nějaké podformule (podstromu) začínající (Qx). Není-li výskyt vázaný, je *volný*. Proměnná je *volná* ve φ, pokud má ve φ volný výskyt, a *vázaná* ve φ, pokud má ve φ vázaný výskyt. Zápis $\varphi(x_1, \ldots, x_n)$ znamená, že $x_1, \ldots, x_n$ jsou všechny volné proměnné ve formuli φ.

Příklad 6.3.7. Proměnná může být volná i vázaná, např. ve formuli $\varphi = (\forall x)(\exists y)(x \leq y) \vee x \leq z$ je první výskyt x vázaný a druhý výskyt volný. (Nakreslete si strom formule!) Proměnná y je vázaná (její jediný výskyt je vázaný) a z je volná. Můžeme tedy psát $\varphi(x, z)$.

Poznámka 6.3.8. Jak uvidíme níže, význam (*pravdivostní hodnota*) formule závisí pouze na ohodnocení volných proměnných. Proměnné v kvantifikátorech, spolu s příslušnými vázanými výskyty, můžeme libovolně přejmenovat.

Otevřené a uzavřené formule

Často budeme mluvit o následujících dvou důležitých druzích formulí:

Definice 6.3.9 (Otevřená a uzavřená formule)**.** Formule je *otevřená*, neobsahuje-li žádný kvantifikátor, a *uzavřená* (neboli *sentence*), pokud nemá žádnou volnou proměnnou

Příklad 6.3.10. Uveďme několik příkladů:

- formule $x + y \leq 0$ je otevřená,
- formule $(\forall x)(\forall y)(x + y \leq 0)$ je uzavřená (tedy je to sentence),
- formule $(\forall x)(x + y \leq 0)$ není ani otevřená, ani uzavřená,
- formule $(0 + 1 = 1) \wedge (1 + 1 = 0)$ je otevřená i uzavřená.

[17] Proměnná x se tedy *nevyskytuje* v symbolu pro kvantifikátor (Qx).

Každá atomická formule je otevřená, otevřené formule jsou jen kombinace atomických pomocí logických spojek. Formule může být otevřená i uzavřená zároveň, v tom případě jsou všechny její termy konstantní. Formule je uzavřená, právě když nemá žádnou volnou proměnnou.[18]

Poznámka 6.3.11. Jak uvidíme později, *pravdivostní hodnota* formule závisí jen na ohodnocení jejích volných proměnných. Speciálně, sentence má v dané struktuře pravdivostní hodnotu 0 nebo 1 (nezávisle na ohodnocení proměnných). To je důvod, proč hrají sentence v logice důležitou roli.

6.3.4 Instance a varianty

Jak jsme viděli, jedna proměnná se může ve formuli vyskytovat v různých 'rolích'. Jde o velmi podobný princip jako v programování, kde jeden identifikátor může v programu znamenat různé proměnné (buď lokální, nebo globální). Pod pojmem *instance* si představte 'dosazení' (termu) do (globální) proměnné (nebo lépe 'nahrazení' proměnné nějakým výrazem, který ji počítá), a pod pojmem *varianta* 'přejmenování' (lokální) proměnné. Vezměme například formuli $\varphi(x)$:

$$P(x) \wedge (\forall x)(Q(x) \wedge (\exists x)R(x))$$

První výskyt proměnné x je volný, druhý je vázaný kvantifikátorem $(\forall x)$, a třetí je vázaný $(\exists x)$. Pokud 'dosadíme' za proměnnou x term $t = 1+1$, dostáváme *instanci* formule φ, kterou označíme $\varphi(x/t)$:

$$P(1+1) \wedge (\forall x)(Q(x) \wedge (\exists x)R(x))$$

Můžeme také přejmenovat kvantifikátory ve formuli, tak získáme *variantu* formule φ, např.:

$$P(x) \wedge (\forall y)(Q(y) \wedge (\exists z)R(z))$$

Jak víme, kdy a jak toto můžeme provést, abychom zachovali význam, tj. aby instance byla *důsledkem* φ, a varianta byla s φ *ekvivalentní*? To nyní chceme zformalizovat.

Instance

Pokud do formule φ *dosadíme* za volnou proměnnou x term t, požadujeme, aby výsledná formule 'říkala' o t 'totéž', co φ o x.

Příklad 6.3.12. Například formule $\varphi(x) = (\exists y)(x + y = 1)$ říká o x, že 'existuje $1 - x$'. Term $t = 1$ lze dosadit, neboť $\varphi(x/t) = (\exists y)(1 + y = 1)$ říká 'existuje 1-1'. Ale term $t = y$ dosadit nelze, $(\exists y)(y + y = 1)$ říká '1 je dělitelné 2'. Problém spočívá v tom, že term $t = y$ obsahuje proměnnou y, jež bude nově vázaná kvantifikátorem $(\exists y)$. Takové situaci se musíme vyhnout.

Definice 6.3.13 (Substituovatelnost a instance)**.** Term t je *substituovatelný* za proměnnou x ve formuli φ, pokud po simultánním nahrazení všech volných výskytů x ve φ za t nevznikne ve φ žádný vázaný výskyt proměnné z t. V tom případě říkáme vzniklé formuli *instance* φ vzniklá substitucí t za x, a označujeme ji $\varphi(x/t)$.

Poznámka 6.3.14. Všimněte si, že term t *není* substituovatelný za x do φ, právě když x má volný výskyt v nějaké podformuli φ tvaru $(Qy)\psi$ a proměnná y se vyskytuje v t. Speciálně, konstantní termy jsou vždy substituovatelné.

[18] Neplatí ale, že formule je otevřená, pokud nemá žádnou vázanou proměnnou, viz formule $(\forall x)0 = 1$.

Varianty

Potřebujeme-li substituovat term t do formule φ, můžeme to udělat vždy, pokud nejprve přejmenujeme všechny kvantifikované proměnné na zcela nové (tj. takové, které se nevyskytují ani ve φ ani v t), a potom substituujeme t do takto vzniklé *varianty* formule φ.

Definice 6.3.15 (Varianta)**.** Má-li formule φ podformuli tvaru $(Qx)\psi$ a je-li y proměnná, taková, že

- y je substituovatelná za x do ψ a
- y nemá volný výskyt v ψ,

potom nahrazením podformule $(Qx)\psi$ formulí $(Qy)\psi(x/y)$ vznikne *varianta* formule φ v podformuli $(Qx)\psi$. *Varianta* říkáme i výsledku postupné variace ve více podformulích.

Všimněte si, že požadavek na proměnnou y z definice varianty je vždy splněn, pokud se y nevyskytuje ve formuli φ.

Příklad 6.3.16. Mějme formuli $\varphi = (\exists x)(\forall y)(x \leq y)$. Potom:

- $(\exists y)(\forall y)(y \leq y)$ není varianta φ, neboť y není substituovatelná za x do $\psi = (\forall y)(x \leq y)$,
- $(\exists x)(\forall x)(x \leq x)$ není varianta φ, neboť x má volný výskyt v podformuli $\psi = (x \leq y)$,
- $(\exists u)(\forall v)(u \leq v)$ je varianta φ.

Tím jsme uzavřeli výklad o syntaxi, následuje sémantika.

6.4 Sémantika

Než se pustíme do formálnějšího výkladu, shrňme stručně sémantiku, tak jak jsme ji už naznačili v předchozích sekcích:

- modely jsou struktury dané signatury,
- formule platí ve struktuře, pokud platí při každém ohodnocení volných proměnných prvky z domény,
- hodnoty termů se vyhodnocují podle jejich stromů, kde symboly nahradíme jejich interpretacemi (relacemi, funkcemi, a konstantami z domény),
- z hodnot termů získáme pravdivostní hodnoty atomických formulí: je výsledná n-tice v relaci?
- hodnoty složených formulí vyhodnocujeme také podle jejich stromu, přičemž $(\forall x)$ hraje roli 'konjunkce přes všechny prvky' a $(\exists y)$ hraje roli 'disjunkce přes všechny prvky' z domény struktury

Nyní formálněji:

6.4.1 Modely jazyka

Definice 6.4.1 (Model jazyka)**.** *Model jazyka* L, nebo také L-struktura, je libovolná struktura v signatuře jazyka L. *Třídu* všech modelů jazyka označíme M_L.

Poznámka 6.4.2. V definici nehraje roli, zda je jazyk s rovností nebo bez. A proč nemůžeme mluvit o *množině* všech modelů M_L, proč musíme říkat *třída*? Protože doménou struktury může být libovolná neprázdná množina, a 'množina všech množin' neexistuje, je to klasický příklad tzv. vlastní třídy. Třída je *'soubor'* všech množin splňujících danou vlastnost (popsatelnou v *jazyce teorie množin*).

Příklad 6.4.3. Mezi modely jazyka uspořádání $L = \langle \leq \rangle$ patří následující struktury: $\langle \mathbb{N}, \leq \rangle$, $\langle \mathbb{Q}, > \rangle$, libovolný orientovaný graf $G = \langle V, E \rangle$, $\langle \mathcal{P}(X), \subseteq \rangle$. Ale také např. $\langle \mathbb{C}, R^{\mathbb{C}} \rangle$ kde $(z_1, z_2) \in R^{\mathbb{C}}$ právě když $|z_1| = |z_2|$ nebo $\langle \{0,1\}, \emptyset \rangle$, což *nejsou* částečná uspořádání.

6.4.2 Hodnota termu

Mějme term t jazyka $L = \langle \mathcal{R}, \mathcal{F} \rangle$ (s rovností nebo bez), a L-strukturu $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, F^{\mathcal{A}} \rangle$. *Ohodnocení proměnných* v množině A je libovolná funkce $e : \mathrm{Var} \to A$.

Definice 6.4.4 (Hodnota termu)**.** *Hodnota termu* t *ve struktuře* $\mathcal{A}$ *při ohodnocení* e, kterou značíme $t^{\mathcal{A}}[e]$, je dána induktivně:

- $x^{\mathcal{A}}[e] = e(x)$ pro proměnnou $x \in \mathrm{Var}$,
- $c^{\mathcal{A}}[e] = c^{\mathcal{A}}$ pro konstantní symbol $c \in \mathcal{F}$, a
- je-li $t = f(t_1, \ldots, t_n)$ složený term, kde $f \in \mathcal{F}$, potom:

$$t^{\mathcal{A}}[e] = f^{\mathcal{A}}(t_1^{\mathcal{A}}[e], \ldots, t_n^{\mathcal{A}}[e])$$

Poznámka 6.4.5. Všimněte si, že hodnota termu závisí pouze na ohodnocení proměnných vyskytujících se v něm. Speciálně, je-li t konstantní term, jeho hodnota na ohodnocení nezávisí. Obecně, každý term t reprezentuje *termovou funkci* $f_t^{\mathcal{A}} \colon A^k \to A$, kde k je počet proměnných v t, a konstantním termům odpovídají konstantní funkce.

Příklad 6.4.6. Uveďme dva příklady:

- Hodnota termu $-(x \vee \bot) \wedge y$ v Booleově algebře $\underline{\mathcal{P}(\{0,1,2\})}$ při ohodnocení e ve kterém $e(x) = \{0,1\}$ a $e(y) = \{1,2\}$ je $\{2\}$.
- Hodnota termu $x + 1$ ve struktuře $\mathcal{N} = \langle \mathbb{N}, \cdot, 3 \rangle$ jazyka $L = \langle +, 1 \rangle$ při ohodnocení e ve kterém $e(x) = 2$ je $(x+1)^{\mathcal{N}}[e] = 6$.

6.4.3 Pravdivostní hodnota formule

Nyní už jsme připraveni definovat *pravdivostní hodnotu*. Lokálně pro ni zavedeme značení PH.

Definice 6.4.7 (Pravdivostní hodnota)**.** Mějme formuli φ v jazyce L, strukturu $\mathcal{A} \in \mathrm{M}_L$, a ohodnocení proměnných $e : \mathrm{Var} \to A$. *Pravdivostní hodnota* φ *v* $\mathcal{A}$ *při ohodnocení* e*,* $\mathrm{PH}^{\mathcal{A}}(\varphi)[e]$, je definována induktivně podle struktury formule:

Pro atomickou formuli $\varphi = R(t_1, \ldots, t_n)$ máme

$$\mathrm{PH}^{\mathcal{A}}(\varphi)[e] = \begin{cases} 1 & \text{pokud } (t_1^{\mathcal{A}}[e], \ldots, t_n^{\mathcal{A}}[e]) \in R^{\mathcal{A}}, \\ 0 & \text{jinak.} \end{cases}$$

Speciálně, je-li φ tvaru $t_1 = t_2$, potom $\mathrm{PH}^{\mathcal{A}}(\varphi)[e] = 1$ právě když $(t_1^{\mathcal{A}}[e], t_2^{\mathcal{A}}[e]) \in =^{\mathcal{A}}$, kde $=^{\mathcal{A}}$ je identita na A, tj. právě když $t_1^{\mathcal{A}}[e] = t_2^{\mathcal{A}}[e]$ (obě strany rovnosti jsou stejný prvek $a \in A$).

Pravdivostní hodnota negace je definována takto:

$$\mathrm{PH}^{\mathcal{A}}(\neg\varphi)[e] = f_{\neg}(\mathrm{PH}^{\mathcal{A}}(\varphi)[e]) = 1 - \mathrm{PH}^{\mathcal{A}}(\varphi)[e]$$

Obdobně pro binární logické spojky, jsou-li φ, ψ a $\square \in \{\wedge, \vee, \to, \leftrightarrow\}$, potom:

$$\mathrm{PH}^{\mathcal{A}}(\varphi \square \psi)[e] = f_{\square}(\mathrm{PH}^{\mathcal{A}}(\varphi)[e], \mathrm{PH}^{\mathcal{A}}(\psi)[e])$$

Zbývá definovat pravdivostní hodnotu pro kvantifikátory, tj. formule tvaru $(Qx)\varphi$. Budeme potřebovat následující značení: Změníme-li v ohodnocení $e : \mathrm{Var} \to A$ hodnotu pro proměnnou x na a, výsledné ohodnocení zapíšeme jako $e(x/a)$. Platí tedy $e(x/a)(x) = a$. Pravdivostní hodnotu pro $(Qx)\varphi$ definujeme takto:

$$\mathrm{PH}^{\mathcal{A}}((\forall x)\varphi)[e] = \min_{a \in A}(\mathrm{PH}^{\mathcal{A}}(\varphi)[e(x/a)])$$
$$\mathrm{PH}^{\mathcal{A}}((\exists x)\varphi)[e] = \max_{a \in A}(\mathrm{PH}^{\mathcal{A}}(\varphi)[e(x/a)])$$

Tedy v ohodnocení e nastavíme hodnotu proměnné x postupně na všechny prvky $a \in A$ a požadujeme, aby PH byla rovna 1 vždy (v případě $\forall$) nebo alespoň jednou (v případě $\exists$).[19]

Poznámka 6.4.8. Pravdivostní hodnota závisí pouze na ohodnocení volných proměnných. Speciálně, je-li φ sentence, potom její pravdivostní hodnota nezávisí na ohodnocení.

Příklad 6.4.9. Vezměme si uspořádané těleso $\underline{\mathbb{Q}}$. Potom:

- $\mathrm{PH}^{\underline{\mathbb{Q}}}(x \leq 1 \wedge \neg(x \leq 0))[e] = 1$ právě když $e(x) \in (0, 1]$,
- $\mathrm{PH}^{\underline{\mathbb{Q}}}((\forall x)(x \cdot y = y))[e] = 1$ právě když $e(y) = 0$,
- $\mathrm{PH}^{\underline{\mathbb{Q}}}((\exists x)(x \leq 0 \wedge \neg x = 0))[e] = 1$ pro každé ohodnocení e (je to sentence), ale
- $\mathrm{PH}^{\mathcal{A}}((\exists x)(x \leq 0 \wedge \neg x = 0))[e] = 0$ (pro každé e), je-li $\mathcal{A} = \langle \mathbb{N}, +, -, 0, \cdot, 1, \leq \rangle$ se standardními operacemi a nerovností.

6.4.4 Platnost

Na základě pravdivostní hodnoty už můžeme definovat klíčový pojem sémantiky, *platnost*.

Definice 6.4.10 (Platnost ve struktuře)**.** Mějme formuli φ a strukturu $\mathcal{A}$ (ve stejném jazyce).

- Je-li e ohodnocení a $\mathrm{PH}^{\mathcal{A}}(\varphi)[e] = 1$, potom říkáme, že φ *platí v* $\mathcal{A}$ *při ohodnocení* e, a píšeme $\mathcal{A} \models \varphi[e]$. (V opačném případě říkáme, že φ *neplatí v* $\mathcal{A}$ *při ohodnocení* e, a píšeme $\mathcal{A} \not\models \varphi[e]$.)

[19] Připomeňme, že $f_{\wedge}(x, y) = \min(x, y)$ a $f_{\vee}(x, y) = \max(x, y)$. Kvantifikátory tedy hrají roli 'konjunkce' ($\forall$) resp. 'disjunkce' ($\exists$) přes všechny prvky struktury.

- Pokud φ platí v $\mathcal{A}$ při každém ohodnocení $e : \mathrm{Var} \to A$, potom říkáme, že φ *je pravdivá (platí) v* $\mathcal{A}$, a píšeme $\mathcal{A} \models \varphi$.
- Pokud $\mathcal{A} \models \neg\varphi$, tj. φ neplatí v $\mathcal{A}$ při žádném ohodnocení (pro každé e máme $\mathcal{A} \not\models \varphi[e]$), potom je φ *lživá v* $\mathcal{A}$.[20]

Shrňme několik jednoduchých vlastností, nejprve týkajících se platnosti při ohodnocení. Buď $\mathcal{A}$ struktura, φ, ψ formule, a e ohodnocení.

- $\mathcal{A} \models \neg\varphi[e]$ právě když $\mathcal{A} \not\models \varphi[e]$,
- $\mathcal{A} \models (\varphi \wedge \psi)[e]$ právě když $\mathcal{A} \models \varphi[e]$ a $\mathcal{A} \models \psi[e]$,
- $\mathcal{A} \models (\varphi \vee \psi)[e]$ právě když $\mathcal{A} \models \varphi[e]$ nebo $\mathcal{A} \models \psi[e]$,
- $\mathcal{A} \models (\varphi \to \psi)[e]$ právě když platí: jestliže $\mathcal{A} \models \varphi[e]$ potom $\mathcal{A} \models \psi[e]$,
- $\mathcal{A} \models (\varphi \leftrightarrow \psi)[e]$ právě když platí: $\mathcal{A} \models \varphi[e]$ právě když $\mathcal{A} \models \psi[e]$,
- $\mathcal{A} \models (\forall x)\varphi[e]$ právě když $\mathcal{A} \models \varphi[e(x/a)]$ pro všechna $a \in A$,
- $\mathcal{A} \models (\exists x)\varphi[e]$ právě když $\mathcal{A} \models \varphi[e(x/a)]$ pro nějaké $a \in A$.
- Je-li term t substituovatelný za proměnnou x do formule φ, potom

$$\mathcal{A} \models \varphi(x/t)[e] \text{ právě když } \mathcal{A} \models \varphi[e(x/a)] \text{ pro } a = t^{\mathcal{A}}[e].$$

- Je-li ψ varianta φ, potom $\mathcal{A} \models \varphi[e]$ právě když $\mathcal{A} \models \psi[e]$.

Cvičení 6.3. Dokažte podrobně všechny uvedené vlastnosti platnosti při ohodnocení.

A jak je tomu s pojmem pravdivosti (platnosti) ve struktuře?

- Pokud $\mathcal{A} \models \varphi$, potom $\mathcal{A} \not\models \neg\varphi$. Je-li φ sentence, potom platí i opačná implikace (tj. platí 'právě když').
- $\mathcal{A} \models \varphi \wedge \psi$ právě když $\mathcal{A} \models \varphi$ a $\mathcal{A} \models \psi$,
- Pokud $\mathcal{A} \models \varphi$ nebo $\mathcal{A} \models \psi$, potom $\mathcal{A} \models \varphi \vee \psi$. Je-li φ sentence, potom platí i opačná implikace (tj. platí 'právě když').
- $\mathcal{A} \models \varphi$ právě když $\mathcal{A} \models (\forall x)\varphi$.

Generální uzávěr formule $\varphi(x_1, \ldots, x_n)$ (tj. $x_1, \ldots, x_n$ jsou všechny volné proměnné formule φ) je sentence $(\forall x_1) \cdots (\forall x_n)\varphi$. Z posledního bodu plyne, že formule platí ve struktuře, právě když v ní platí její generální uzávěr.

Cvičení 6.4. Dokažte podrobně všechny uvedené vlastnosti platnosti ve struktuře.

Cvičení 6.5. Najděte příklad struktury $\mathcal{A}$ a formule φ takových, že $\mathcal{A} \not\models \varphi$ a zároveň $\mathcal{A} \not\models \neg\varphi$.

Cvičení 6.6. Najděte příklad struktury $\mathcal{A}$ a formulí φ, ψ takových, že $\mathcal{A} \models \varphi \vee \psi$ ale $\mathcal{A} \not\models \varphi$ ani $\mathcal{A} \not\models \psi$.

[20] Pozor, *lživá* není totéž, co *není pravdivá*! To platí jen pro sentence.

6.5 Vlastnosti teorií

Na základě pojmu *platnosti* vybudujeme syntaktickou terminologii obdobně jako ve výrokové logice. *Teorie* jazyka L je libovolná množina T L-formulí, prvkům teorie říkáme *axiomy*. *Model* teorie T je L-struktura, ve které platí všechny axiomy teorie T, tj. $\mathcal{A} \models \varphi$ pro všechna $\varphi \in T$, což značíme $\mathcal{A} \models T$. *Třída modelů*[21] teorie T je:

$$\mathrm{M}_L(T) = \{\mathcal{A} \in \mathrm{M}_L \mid \mathcal{A} \models T\}$$

Stejně jako ve výrokové logice budeme často vynechávat jazyk L, bude-li zřejmý z kontextu, a budeme psát $M(\varphi_1, \ldots, \varphi_n)$ místo $M(\{\varphi_1, \ldots, \varphi_n\})$ a $M(T, \varphi)$ místo $M(T \cup \{\varphi\})$.

6.5.1 Platnost v teorii

Je-li T teorie v jazyce L a φ L-formule, potom říkáme, že φ je:

- *pravdivá (platí) v* T, značíme $T \models \varphi$, pokud $\mathcal{A} \models \varphi$ pro všechna $\mathcal{A} \in \mathrm{M}(T)$ (neboli: $\mathrm{M}(T) \subseteq \mathrm{M}(\varphi)$),
- *lživá v* T, pokud $T \models \neg\varphi$, tj. pokud je lživá v každém modelu T (neboli: $\mathrm{M}(T) \cap \mathrm{M}(\varphi) = \emptyset$),
- *nezávislá v* T, pokud není pravdivá v T ani lživá v T.

Máme-li prázdnou teorii $T = \emptyset$ (tj. $\mathrm{M}(T) = \mathrm{M}_L$), potom teorii T vynecháváme, píšeme $\models \varphi$, a říkáme, že *φ je pravdivá (v logice), (logicky) platí, je tautologie*; podobně pro ostatní pojmy.

Teorie je *sporná*, jestliže v ní platí *spor* $\bot$, který v predikátové logice můžeme definovat jako $R(x_1, \ldots, x_n) \wedge \neg R(x_1, \ldots, x_n)$, kde R je libovolný (třeba první) relační symbol z jazyka nebo rovnost (nemá-li jazyk relační symbol, musí být s rovností). Teorie je *sporná*, právě když v ní platí každá formule, nebo, ekvivalentně, právě když nemá žádný model. Jinak říkáme, že je teorie *bezesporná* (neplatí-li v ní spor, ekvivalentně má-li alespoň jeden model).

Sentencím pravdivým v T říkáme *důsledky* T; *množina všech důsledků* T v jazyce L je:

$$\mathrm{Csq}_L(T) = \{\varphi \mid \varphi \text{ je sentence a } T \models \varphi\}$$

Kompletnost v predikátové logice

Jak je tomu s pojmem *kompletnosti* teorie?[22]

Definice 6.5.1. Teorie je *kompletní*, je-li bezesporná a každá *sentence* je v ní buď pravdivá, nebo lživá.

Nemůžeme ale říci, že je teorie kompletní, právě když má jediný model. Máme-li totiž jeden model, dostáváme z něj nekonečně mnoho jiných, ale *izomorfních* modelů, tj. lišících se jen pojmenováním prvků univerza.[23] Uvažovat jediný model 'až na izomorfismus' by ale nebylo dostatečné. Správným pojmem je tzv. *elementární ekvivalence*:

[21] Připomeňme, že nemůžeme říkat 'množina'.

[22] Připomeňme, že *výroková* teorie je kompletní, je-li bezesporná a každý výrok v ní buď platí, nebo platí jeho negace. Ekvivalentně, má právě jeden model.

[23] Formálně pojem *izomorfismu* definujeme později v části o *teorii modelů*, v Sekci 9.2, jde ale o zobecnění izomorfismu který znáte z teorie grafů.

Definice 6.5.2. Struktury $\mathcal{A}, \mathcal{B}$ (v témž jazyce) jsou *elementárně ekvivalentní*, pokud v nich platí tytéž sentence. Značíme $\mathcal{A} \equiv \mathcal{B}$.

Příklad 6.5.3. Příkladem struktur, které jsou elementárně ekvivalentní, ale ne izomorfní, jsou uspořádané množiny $\mathcal{A} = \langle \mathbb{Q}, \leq \rangle$ a $\mathcal{B} = \langle \mathbb{R}, \leq \rangle$. Izomorfní nejsou proto, že $\mathbb{Q}$ je spočetná zatímco $\mathbb{R}$ nespočetná množina, neexistuje tedy dokonce žádná *bijekce* mezi jejich univerzy. Není těžké ukázat, že pro každou sentenci φ platí $\mathcal{A} \models \varphi \Leftrightarrow \mathcal{B} \models \varphi$: indukcí podle struktury formule φ, jediný netriviální případ je existenční kvantifikátor, a klíčovou vlastností je *hustota* obou uspořádání, tj. následující vlastnost:

$$(x \leq y \wedge \neg x = y) \to (\exists z)(x \leq z \wedge z \leq y \wedge \neg x = z \wedge \neg y = z)$$

Pozorování 6.5.4. *Teorie je kompletní, právě když má právě jeden model až na elementární ekvivalenci.*

Platnost pomocí nesplnitelnosti

Otázku pravdivosti (platnosti) v dané teorii lze převést na problém existence modelu:

Tvrzení 6.5.5 (O nesplnitelnosti a pravdivosti)**.** *Je-li T teorie a φ* sentence *(ve stejném jazyce), potom platí: $T \cup \{\neg\varphi\}$ nemá model, právě když $T \models \varphi$.*

Důkaz. Platí následující ekvivalence: $T \cup \{\neg\varphi\}$ nemá model, právě když $\neg\varphi$ neplatí v žádném modelu T, právě když (neboť je to sentence) φ platí v každém modelu T. □

Předpoklad, že φ je sentence, je nutný: uvažte teorii $T = \{P(c)\}$ a formuli $\varphi = P(x)$ (což není sentence). Potom $\{P(c), \neg P(x)\}$ nemá model, ale $P(c) \not\models P(x)$. (Zde P je unární relační, a c konstantní symbol.)

6.5.2 Příklady teorií

Uveďme několik příkladů důležitých teorií.

Teorie grafů

Teorie grafů je teorie v jazyce $L = \langle E \rangle$ s rovností, splňující axiomy *ireflexivity* a *symetrie*:

$$T_{\text{graph}} = \{\neg E(x,x), E(x,y) \to E(y,x)\}$$

Modely T_{graph} jsou struktury $\mathcal{G} = \langle G, E^{\mathcal{G}} \rangle$, kde $E^{\mathcal{G}}$ je symetrická ireflexivní relace, jde tedy o tzv. *jednoduché* grafy, kde hranu $\{x, y\}$ reprezentuje dvojice uspořádaných hran $(x, y), (y, x)$.

- Formule $\neg x = y \to E(x, y)$ platí v grafu, právě když je *úplný*. Je tedy nezávislá v T_{graph}.
- Formule $(\exists y_1)(\exists y_2)(\neg y_1 = y_2 \wedge E(x, y_1) \wedge E(x, y_2) \wedge (\forall z)(E(x, z) \to z = y_1 \vee z = y_2)$ vyjadřuje, že každý vrchol má stupeň právě 2. Platí tedy právě v grafech, které jsou disjunktní sjednocení kružnic, a je nezávislá v teorii T_{graph}.

Teorie uspořádání

Teorie uspořádání je teorie v jazyce uspořádání $L = \langle \leq \rangle$ s rovností, jejíž axiomy jsou:

$$\begin{aligned} T = \{ & x \leq x, \\ & x \leq y \wedge y \leq x \to x = y, \\ & x \leq y \wedge y \leq z \to x \leq z \} \end{aligned}$$

Těmto axiomům říkáme *reflexivita*, *antisymetrie*, *tranzitivita*. Modely T jsou L-struktury $\langle S, \leq^S \rangle$, ve kterých platí axiomy T, tzv. *(částečně) uspořádané množiny*. Např: $\mathcal{A} = \langle \mathbb{N}, \leq \rangle$, $\mathcal{B} = \langle \mathcal{P}(X), \subseteq \rangle$ pro $X = \{0, 1, 2\}$.

- Formule $x \leq y \vee y \leq x$ (*linearita*) platí v $\mathcal{A}$, ale neplatí v $\mathcal{B}$, neboť neplatí např. při ohodnocení kde $e(x) = \{0\}$, $e(y) = \{1\}$ (píšeme $\mathcal{B} \not\models \varphi[e]$). Je tedy nezávislá v T.
- Sentence $(\exists x)(\forall y)(y \leq x)$ (označme ji ψ) je pravdivá v $\mathcal{B}$ a lživá v $\mathcal{A}$, píšeme $\mathcal{B} \models \psi$, $\mathcal{A} \models \neg\psi$. Je tedy také nezávislá v T.
- Formule $(x \leq y \wedge y \leq z \wedge z \leq x) \to (x = y \wedge y = z)$ (označme ji χ) je pravdivá v T, píšeme $T \models \chi$. Totéž platí pro její *generální uzávěr* $(\forall x)(\forall y)(\forall z)\chi$.

Algebraické teorie

- *Teorie grup* je teorie v jazyce $L = \langle +, -, 0 \rangle$ s rovností, jejíž axiomy jsou:

$$\begin{aligned} T_1 = \{ & x + (y + z) = (x + y) + z, \\ & 0 + x = x,\ x + 0 = 0, \\ & x + (-x) = 0,\ (-x) + x = 0 \} \end{aligned}$$

  Těmto vlastnostem říkáme *asociativita* $+$, *neutralita* 0 *vůči* $+$, a $-x$ *je inverzní prvek k* x *(vůči* $+$ *a* 0*)*.

- *Teorie komutativních grup* má navíc axiom $x + y = y + x$ (*komutativita* $+$), je tedy:

$$T_2 = T_1 \cup \{x + y = y + x\}$$

- *Teorie okruhů* je v jazyce $L = \langle +, -, 0, \cdot, 1 \rangle$ s rovností, má navíc axiomy:

$$\begin{aligned} T_3 = T_2 \cup \{ & 1 \cdot x = x \cdot 1, \\ & x \cdot (y \cdot z) = (x \cdot y) \cdot z, \\ & x \cdot (y + z) = x \cdot y + x \cdot z, \\ & (x + y) \cdot z = x \cdot z + y \cdot z \} \end{aligned}$$

  Těmto vlastnostem říkáme *neutralita* 1 *vůči* $\cdot$, *asociativita* $\cdot$, a *(levá i pravá) distributivita* $\cdot$ *vůči* $+$.

- *Teorie komutativních okruhů* má navíc axiom *komutativity* $\cdot$, máme tedy:

$$T_4 = T_3 \cup \{x \cdot y = y \cdot x\}$$

- *Teorie těles* je ve stejném jazyce, ale má navíc axiomy *existence inverzního prvku* $k \cdot$ a *netriviality*:

$$T_5 = T_4 \cup \{\neg\, x = 0 \to (\exists y)(x \cdot y = 1), \neg\, 0 = 1\}$$

- *Teorie uspořádaných těles* je v jazyce $\langle +, -, 0, \cdot, 1, \leq \rangle$ s rovností, sestává z axiomů teorie těles, teorie uspořádání spolu s axiomem linearity, a z následujících axiomů *kompatibility uspořádání*: $x \leq y \to (x + z \leq y + z)$ a $(0 \leq x \wedge 0 \leq y) \to 0 \leq x \cdot y$. (Modely jsou tedy tělesa s *lineárním (totálním)* uspořádáním, které je kompatibilní s tělesovými operacemi v tomto smyslu.)

6.6 Podstruktura, expanze, redukt

V této sekci se podíváme na způsoby, jak můžeme vytvářet nové struktury z existujících.

Podstruktura

Pojem *podstruktury* zobecňuje podgrupy, podprostory vektorového prostoru, a indukované podgrafy grafu: vybereme nějakou podmnožinu B univerza struktury $\mathcal{A}$, a vytvoříme na ní strukturu $\mathcal{B}$ stejné signatury, která 'zdědí' relace, funkce, a konstanty. Abychom to mohli provést, potřebujeme, aby byla množina B *uzavřená* na všechny funkce a obsahovala všechny konstanty.[24]

Definice 6.6.1 (Podstruktura)**.** Mějme strukturu $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}} \rangle$ v signatuře $\langle \mathcal{R}, \mathcal{F} \rangle$. Struktura $\mathcal{B} = \langle B, \mathcal{R}^{\mathcal{B}}, \mathcal{F}^{\mathcal{B}} \rangle$ je *(indukovaná) podstruktura* $\mathcal{A}$, značíme $\mathcal{B} \subseteq \mathcal{A}$, jestliže

- $\emptyset \neq B \subseteq A$,
- $R^{\mathcal{B}} = R^{\mathcal{A}} \cap B^{\mathrm{ar(R)}}$ pro každý relační symbol $R \in \mathcal{R}$,
- $f^{\mathcal{B}} = f^{\mathcal{A}} \cap (B^{\mathrm{ar(f)}} \times B)$ pro každý funkční symbol $f \in \mathcal{F}$ (tj. funkce $f^{\mathcal{B}}$ je restrikce $f^{\mathcal{A}}$ na množinu B, a její výstupy jsou všechny také z B),
- speciálně, pro každý konstantní symbol $c \in \mathcal{F}$ máme $c^{\mathcal{B}} = c^{\mathcal{A}} \in B$.

Množina $C \subseteq A$ je *uzavřená* na funkci $f : A^n \to A$, pokud $f(x_1, \ldots, x_n) \in C$ pro všechna $x_i \in C$. Platí:

Pozorování 6.6.2. *Množina $\emptyset \neq C \subseteq A$ je univerzem podstruktury struktury $\mathcal{A}$, právě když je C uzavřená na všechny funkce struktury $\mathcal{A}$ (včetně konstant).*

V tom případě říkáme této podstruktuře *restrikce* $\mathcal{A}$ na množinu C, a značíme ji $\mathcal{A} \restriction C$.

Příklad 6.6.3. $\underline{\mathbb{Z}} = \langle \mathbb{Z}, +, \cdot, 0 \rangle$ je podstrukturou $\underline{\mathbb{Q}} = \langle \mathbb{Q}, +, \cdot, 0 \rangle$, můžeme psát $\underline{\mathbb{Z}} = \underline{\mathbb{Q}} \restriction \mathbb{Z}$. Struktura $\underline{\mathbb{N}} = \langle \mathbb{N}, +, \cdot, 0 \rangle$ je podstrukturou obou těchto struktur, $\underline{\mathbb{N}} = \underline{\mathbb{Q}} \restriction \mathbb{N} = \underline{\mathbb{Z}} \restriction \mathbb{N}$.

[24] Stejně jako ne každá množina vektorů je podprostor, k tomu musí obsahovat nulový vektor, ke každému vektoru obsahovat všechny jeho skalární násobky, a pro každou dvojici vektorů obsahovat jejich součet. Jinými slovy, jen (neprázdné) množiny uzavřené na *lineární kombinace* vektorů dávají vzniknout podprostorům.

Platnost v podstruktuře

Jak je tomu s platností formulí v podstruktuře? Uveďme několik jednoduchých pozorování o *otevřených* formulích.

Pozorování 6.6.4. *Je-li $\mathcal{B} \subseteq \mathcal{A}$, potom pro každou* otevřenou *formuli φ a ohodnocení proměnných $e\colon \mathrm{Var} \to B$ platí: $\mathcal{B} \models \varphi[e]$ právě když $\mathcal{A} \models \varphi[e]$.*

Důkaz. Pro atomické formule je zřejmé, dále snadno dokážeme indukcí podle struktury formule. □

Důsledek 6.6.5. Otevřená *formule platí ve struktuře $\mathcal{A}$, právě když platí v každé podstruktuře $\mathcal{B} \subseteq \mathcal{A}$.*

Říkáme, že teorie T je *otevřená*, jsou-li všechny její axiomy otevřené formule.

Důsledek 6.6.6. *Modely otevřené teorie jsou uzavřené na podstruktury, tj. každá podstruktura modelu otevřené teorie je také model této teorie.*

Příklad 6.6.7. Teorie grafů je otevřená. Každá podstruktura grafu (modelu teorie grafů) je také graf, říkáme mu (indukovaný) *podgraf*.[25] Podobně např. pro podgrupy nebo Booleovy podalgebry.

Příklad 6.6.8. Teorie těles není otevřená. Jak si ukážeme později, není dokonce ani *otevřeně axiomatizovatelná*, tj. neexistuje jí ekvivalentní otevřená teorie – kvantifikátoru v axiomu o existenci inverzního prvku se nelze nijak zbavit. Podstruktura tělesa reálných čísel $\mathbb{Q}$ na množině všech celých čísel $\mathbb{Q} \restriction \mathbb{Z}$ není těleso. (Je to tzv. *okruh*, ale nenulové prvky kromě $1, -1$ nemají multiplikativní inverz, např. rovnice $2 \cdot x = 1$ nemá v $\mathbb{Z}$ řešení).

Generovaná podstruktura

Co dělat, máme-li podmnožinu univerza, která *není* uzavřená na funkce struktury? V tom případě uvážíme *uzávěr* této množiny na funkce.[26]

Definice 6.6.9. Mějme strukturu $\mathcal{A} = \langle A, \mathcal{R}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}} \rangle$ a neprázdnou podmnožinu $X \subseteq A$. Označme jako B nejmenší podmnožinu A, která obsahuje množinu X a je uzavřená na všechny funkce struktury $\mathcal{A}$ (tj. také obsahuje všechny konstanty). Potom o podstruktuře $\mathcal{A} \restriction B$ říkáme, že je *generovaná* množinou X, a značíme ji $\mathcal{A}\langle X \rangle$.

Příklad 6.6.10. Uvažme struktury $\underline{\mathbb{Q}} = \langle \mathbb{Q}, +, \cdot, 0 \rangle$, $\underline{\mathbb{Z}} = \langle \mathbb{Z}, +, \cdot, 0 \rangle$, a $\underline{\mathbb{N}} = \langle \mathbb{N}, +, \cdot, 0 \rangle$. Potom $\underline{\mathbb{Q}}\langle\{1\}\rangle = \underline{\mathbb{N}}$, $\underline{\mathbb{Q}}\langle\{-1\}\rangle = \underline{\mathbb{Z}}$, a $\underline{\mathbb{Q}}\langle\{2\}\rangle$ je podstruktura $\underline{\mathbb{N}}$ na množině všech sudých čísel.

Příklad 6.6.11. Pokud $\mathcal{A}$ nemá žádné funkce (ani konstanty), např. je-li to graf či uspořádání, potom není čím generovat, a $\mathcal{A}\langle X \rangle = \mathcal{A} \restriction X$.

[25] Samotný pojem *podgraf* v teorii grafů často znamená jen $E^{\mathcal{B}} \subseteq B \times B$, nikoliv $E^{\mathcal{B}} = B \times B$. My ale budeme používat slovo *podgraf* ve striktnějším smyslu, jako indukovaný podgraf.

[26] Viz pojem *lineárního obalu* množiny vektorů.

Expanze a redukt

Prozatím jsme konstruovali nové struktury změnou univerza. Můžeme ale také nechat univerzum stejné, a přidat resp. odebrat relace, funkce, a konstanty. Výsledku takové operace říkáme *expanze* resp. *redukt*. Všimněte si, že jde o strukturu v jiné signatuře.

Definice 6.6.12 (Expanze a redukt)**.** Mějme jazyky $L \subseteq L'$, L-strukturu $\mathcal{A}$, a L'-strukturu $\mathcal{A}'$ na stejné doméně $A = A'$. Jestliže je interpretace každého symbolu z L [relačního, funkčního, konstantního] stejná [relace, funkce, konstanta] v $\mathcal{A}$ i v $\mathcal{A}'$ potom říkáme, že struktura $\mathcal{A}'$ je *expanzí* struktury $\mathcal{A}$ do jazyka L' (také říkáme, že je L'*-expanzí*) a že struktura $\mathcal{A}$ je *reduktem* struktury $\mathcal{A}'$ na jazyk L (také říkáme, že je L*-reduktem*).

Příklad 6.6.13. Mějme grupu celých čísel $\langle \mathbb{Z}, +, -, 0\rangle$. Potom struktura $\langle \mathbb{Z}, +\rangle$ je jejím reduktem, zatímco struktura $\langle \mathbb{Z}, +, -, 0, \cdot, 1\rangle$ (*okruh* celých čísel) je její expanzí.

Příklad 6.6.14. Mějme graf $\mathcal{G} = \langle G, E^{\mathcal{G}}\rangle$. Potom struktura $\langle G, E^G, c_v^{\mathcal{G}}\rangle_{v\in G}$ v jazyce $\langle E, c_v\rangle_{v\in G}$, kde $c_v^{\mathcal{G}} = v$ pro všechny vrcholy $v \in G$, je *expanzí* $\mathcal{G}$ *o jména prvků (z množiny* G*).*

6.6.1 Věta o konstantách

Věta o konstantách říká (neformálně), že splnit formuli s volnou proměnnou je totéž, co splnit sentenci, ve které je tato volná proměnná nahrazena (substituována) *novým* konstantním symbolem (který není nijak svázaný žádnými axiomy). Klíčem je fakt, že tento nový symbol může být v modelech interpretován jako libovolný (tj. každý) prvek. Tento trik později využijeme v tablo metodě.

Věta 6.6.15 (O konstantách)**.** *Mějme formuli φ v jazyce L s volnými proměnnými $x_1, \ldots, x_n$. Označme L' rozšíření jazyka o nové konstantní symboly $c_1, \ldots, c_n$ a buď T' stejná teorie jako T ale v jazyce L'. Potom platí:*

$$T \models \varphi \text{ právě když } T' \models \varphi(x_1/c_1, \ldots, x_n/c_n)$$

Důkaz. Tvrzení stačí dokázat pro jednu volnou proměnnou x a jednu konstantu c, indukcí se snadno rozšíří na n konstant.

Předpokládejme nejprve, že φ platí v každém modelu teorie T. Chceme ukázat, že $\varphi(x/c)$ platí v každém modelu $\mathcal{A}'$ teorie T'. Vezměme tedy takový model $\mathcal{A}'$ a libovolné ohodnocení $e\colon \mathrm{Var} \to A'$ a ukažme, že $\mathcal{A}' \models \varphi(x/c)[e]$.

Označme jako $\mathcal{A}$ redukt $\mathcal{A}'$ na jazyk L ('zapomeneme' konstantu $c^{\mathcal{A}'}$). Všimněte si, že $\mathcal{A}$ je model teorie T (axiomy T jsou tytéž jako T', neobsahují symbol c) tedy v něm platí φ. Protože dle předpokladu platí $\mathcal{A} \models \varphi[e']$ pro *libovolné* ohodnocení e', platí i pro ohodnocení $e(x/c^{\mathcal{A}'})$ ve kterém ohodnotíme proměnnou x interpretací konstantního symbolu c ve struktuře $\mathcal{A}'$, máme tedy $\mathcal{A} \models \varphi[e(x/c^{\mathcal{A}'})]$. To ale znamená, že $\mathcal{A}' \models \varphi(x/c)[e]$, což jsme chtěli dokázat.

Naopak, předpokládejme, že $\varphi(x/c)$ platí v každém modelu teorie T' a ukažme, že φ platí v každém modelu $\mathcal{A}$ teorie T. Zvolme tedy takový model $\mathcal{A}$ a nějaké ohodnocení $e\colon \mathrm{Var} \to A$ a ukažme, že $\mathcal{A} \models \varphi[e]$.

Označme jako $\mathcal{A}'$ expanzi $\mathcal{A}$ do jazyka L', kde konstantní symbol c interpretujeme jako prvek $c^{\mathcal{A}'} = e(x)$. Protože dle předpokladu platí $\mathcal{A}' \models \varphi(x/c)[e']$ pro všechna ohodnocení e', platí i $\mathcal{A}' \models \varphi(x/c)[e]$, což ale znamená, že $\mathcal{A}' \models \varphi[e]$. (Neboť $e = e(x/c^{\mathcal{A}'})$ a $\mathcal{A}' \models \varphi(x/c)[e]$ platí právě když $\mathcal{A}' \models \varphi[e(x/c^{\mathcal{A}'})]$, což je $\mathcal{A}' \models \varphi[e]$.) Formule φ ale neobsahuje c (zde používáme, že c je *nový*), máme tedy i $\mathcal{A} \models \varphi[e]$. □

6.7 Extenze teorií

Pojem *extenze* teorie definujeme stejně jako ve výrokové logice:

Definice 6.7.1 (Extenze teorie)**.** Mějme teorii T v jazyce L.

- *Extenze* teorie T je libovolná teorie T' v jazyce $L' \supseteq L$ splňující $\mathrm{Csq}_L(T) \subseteq \mathrm{Csq}_{L'}(T')$,
- je to *jednoduchá extenze*, pokud $L' = L$,
- je to *konzervativní extenze*, pokud $\mathrm{Csq}_L(T) = \mathrm{Csq}_L(T') = \mathrm{Csq}_{L'}(T') \cap \mathrm{Fm}_L$, kde Fm_L značí množinu všech formulí v jazyce L.
- Teorie T' (v jazyce L) je *ekvivalentní* teorii T, pokud je T' extenzí T a T extenzí T'.

Podobně jako ve výrokové logice, pro teorie ve stejném jazyce platí následující sémantický popis těchto pojmů:

Pozorování 6.7.2. *Mějme teorie T, T' v jazyce L. Potom:*

- *T' je extenze T, právě když $\mathrm{M}_L(T') \subseteq \mathrm{M}_L(T)$.*
- *T' je ekvivalentní s T, právě když $\mathrm{M}_L(T') = \mathrm{M}_L(T)$.*

Jak je tomu v případě, kdy teorie T' je nad větším jazykem než T? Připomeňme situaci ve výrokové logice, popsanou v Pozorování 2.4.7. Zformulujeme a dokážeme analogické tvrzení: Zatímco ve výrokové logice jsme přidávali hodnoty pro nové prvovýroky, resp. je zapomínali, v predikátové logice budeme expandovat resp. redukovat struktury, tj. přidávat nebo zapomínat interpretace relačních, funkčních, a konstantních symbolů. Princip tvrzení ale zůstává stejný.

Tvrzení 6.7.3. *Mějme jazyky $L \subseteq L'$, teorii T v jazyce L, a teorii T' v jazyce L'.*

(i) T' je extenzí teorie T, právě když redukt každého modelu T' na jazyk L je modelem T.

(ii) Pokud je T' extenzí teorie T, a každý model T lze expandovat do jazyka L' na nějaký model teorie T', potom je T' konzervativní extenzí teorie T.

Poznámka 6.7.4. V části (ii) platí i opačná implikace, důkaz ale není tak jednoduchý, jako ve výrokové logice, a proto ho neuvedeme. (Problémem je jak získat z modelu T který nelze expandovat na model T' L-sentenci, která platí v T ale ne v T'.)

Důkaz. Nejprve dokažme (i): Mějme model $\mathcal{A}'$ teorie T' a označme jako $\mathcal{A}$ jeho redukt na jazyk L. Protože T' je extenzí teorie T, platí v T', a tedy i v $\mathcal{A}'$, každý axiom $\varphi \in T$. Potom ale i $\mathcal{A} \models \varphi$ (φ obsahuje jen symboly z jazyka L), tedy $\mathcal{A}$ je modelem T.

Na druhou stranu, mějme L-sentenci φ takovou, že $T \models \varphi$. Chceme ukázat, že $T' \models \varphi$. Pro libovolný model $\mathcal{A}' \in \mathrm{M}_{L'}(T')$ víme, že jeho L-redukt $\mathcal{A}$ je modelem T, tedy $\mathcal{A} \models \varphi$. Z toho plyne i $\mathcal{A}' \models \varphi$ (opět proto, že φ je v jazyce L).

Nyní (ii): Vezměme libovolnou L-sentenci φ, která platí v teorii T', a ukažme, že platí i v T. Každý model $\mathcal{A}$ teorie T lze expandovat na nějaký model $\mathcal{A}'$ teorie T'. Víme, že $\mathcal{A}' \models \varphi$, takže i $\mathcal{A} \models \varphi$. Tím jsme dokázali, že $T \models \varphi$, tj. jde o konzervativní extenzi.

□

6.7.1 Extenze o definice

Nyní si ukážeme speciální druh konzervativní extenze, tzv. extenzi *o definice* nových (relačních, funkčních, konstantních) symbolů.

Definice relačního symbolu

Nejjednodušším případem je definování nového relačního symbolu $R(x_1, \ldots, x_n)$. Jako definice může sloužit libovolná formule s n volnými proměnnými $\psi(x_1, \ldots, x_n)$.

Příklad 6.7.5. Uveďme nejprve několik příkladů:

- Jakoukoliv teorii v jazyce s rovností můžeme rozšířit o binární relační symbol $\neq$, který *definujeme* formulí $\neg x_1 = x_2$. To znamená, že požadujeme, aby platilo: $x_1 \neq x_2 \leftrightarrow \neg x_1 = x_2$.
- Teorii uspořádání můžeme rozšířit o symbol $<$ pro ostré uspořádání, který *definujeme* formulí $x_1 \leq x_2 \wedge \neg x_1 = x_2$. To znamená, že požadujeme, aby platilo $x_1 < x_2 \leftrightarrow x_1 \leq x_2 \wedge \neg x_1 = x_2$.
- V aritmetice můžeme zavést symbol $\leq$, pomocí $x_1 \leq x_2 \leftrightarrow (\exists y)(x_1 + y = x_2)$.

Nyní uvedeme definici:

Definice 6.7.6 (Definice relačního symbolu)**.** Mějme teorii T a formuli $\psi(x_1, \ldots, x_n)$ v jazyce L. Označme jako L' rozšíření jazyka L o nový n-ární relační symbol R. *Extenze teorie T o definici R formulí ψ* je L'-teorie:

$$T' = T \cup \{R(x_1, \ldots, x_n) \ \leftrightarrow \ \psi(x_1, \ldots, x_n)\}$$

Všimněte si, že každý model T lze *jednoznačně* expandovat na model T'. Z Tvrzení 6.7.3 potom ihned plyne následující:

Důsledek 6.7.7. *T' je konzervativní extenze T.*

Ukážeme si ještě, že nový symbol lze ve formulích nahradit jeho definicí, a získat tak (T'-ekvivalentní) formuli v původním jazyce:

Tvrzení 6.7.8. *Pro každou L'-formuli φ' existuje L-formule φ taková, že $T' \models \varphi' \leftrightarrow \varphi$.*

Důkaz. Je třeba nahradit atomické podformule s novým symbolem R, tj. tvaru $R(t_1, \ldots, t_n)$. Takovou podformuli nahradíme formulí $\psi'(x_1/t_1, \ldots, x_n/t_n)$, kde ψ' je varianta ψ zaručující substituovatelnost všech termů, tj. například přejmenujeme všechny vázané proměnné ψ na zcela nové (nevyskytující se ve formuli φ'). □

Definice funkčního symbolu

Nový funkční symbol definujeme obdobným způsobem, musíme si ale být jisti, že definice dává jednoznačnou možnost, jak nový symbol interpretovat.

Příklad 6.7.9. Opět začneme příklady:

- V teorii grup můžeme zavést *binární* funkční symbol $-_b$ pomocí $+$ a unárního $-$ takto:

$$x_1 -_b x_2 = y \;\leftrightarrow\; x_1 + (-x_2) = y$$

  Je zřejmé, že pro každá x, y *existuje jednoznačné* z splňující definici.

- Uvažme teorii *lineárních uspořádání*, tj. teorii uspořádání spolu s axiomem linearity $x \leq y \vee y \leq x$. Definujme binární funkční symbol min takto:

$$\min(x_1, x_2) = y \;\leftrightarrow\; y \leq x_1 \wedge y \leq x_2 \wedge (\forall z)(z \leq x_1 \wedge z \leq x_2 \to z \leq y)$$

  Existence a jednoznačnost platí díky linearitě. Pokud bychom ale měli pouze teorii uspořádání, taková formule by nebyla dobrou definicí: v některých modelech by $\min(x_1, x_2)$ pro některé prvky neexistovalo, selhala by tedy požadovaná *existence*.

Definice 6.7.10 (Definice funkčního symbolu)**.** Mějme teorii T a formuli $\psi(x_1, \ldots, x_n, y)$ v jazyce L. Označme jako L' rozšíření jazyka L o nový n-ární funkční symbol f. Nechť v teorii T platí:

- *axiom existence* $(\exists y)\psi(x_1, \ldots, x_n, y)$,
- *axiom jednoznačnosti* $\psi(x_1, \ldots, x_n, y) \wedge \psi(x_1, \ldots, x_n, z) \to y = z$.

Potom *extenze teorie* T *o definici* f *formulí* ψ je L'-teorie:

$$T' = T \cup \{f(x_1, \ldots, x_n) = y \;\leftrightarrow\; \psi(x_1, \ldots, x_n, y)\}$$

Formule ψ tedy definuje v každém modelu $(n+1)$-ární relaci, a po této relaci požadujeme, aby byla funkcí, tj. aby pro každou n-tici prvků existovala jednoznačná možnost, jak ji rozšířit do $(n+1)$-tice, která je prvkem této relace. Všimněte si, že je-li definující formule ψ tvaru $t(x_1, \ldots, x_n) = y$, kde $x_1, \ldots, x_n$ jsou proměnné L-termu t, potom axiomy existence a jednoznačnosti vždy platí.

Opět platí, že každý model T lze *jednoznačně* expandovat na model T', tedy:

Důsledek 6.7.11. *T' je konzervativní extenze T.*

A platí také analogické tvrzení o rozvádění definic:

Tvrzení 6.7.12. *Pro každou L'-formuli φ' existuje L-formule φ taková, že $T' \models \varphi' \leftrightarrow \varphi$.*

Důkaz. Stačí dokázat pro formuli φ' s jediným výskytem symbolu f; je-li výskytů více, aplikujeme postup induktivně, v případě vnořených výskytů v jednom termu $f(\ldots f(\ldots)\ldots)$ postupujeme od vnitřních k vnějším.

Označme φ^* formuli vzniklou z φ' nahrazením termu $f(t_1, \ldots, t_n)$ *novou* proměnnou z. Formuli φ zkonstruujeme takto:

$$(\exists z)(\varphi^* \wedge \psi'(x_1/t_1, \ldots, x_n/t_n, y/z))$$

kde ψ' je varianta ψ zaručující substituovatelnost všech termů.

Mějme model $\mathcal{A}$ teorie T' a ohodnocení e. Označme $a = (f(t_1, \ldots, t_n))^{\mathcal{A}}[e]$. Díky existenci a jednoznačnosti platí:

$$\mathcal{A} \models \psi'(x_1/t_1, \ldots, x_n/t_n, y/z)[e] \;\text{ právě když }\; e(z) = a$$

Máme tedy $\mathcal{A} \models \varphi[e]$, právě když $\mathcal{A} \models \varphi^*[e(z/a)]$, právě když $\mathcal{A} \models \varphi'[e]$. To platí pro libovolné ohodnocení e, tedy $\mathcal{A} \models \varphi' \leftrightarrow \varphi$ pro každý model T', tedy $T' \models \varphi' \leftrightarrow \varphi$. □

Definice konstantního symbolu

Konstantní symbol je speciálním případem funkčního symbolu arity 0. Platí tedy stejná tvrzení. Axiomy existence a jednoznačnosti jsou: $(\exists y)\psi(y)$ a $\psi(y) \wedge \psi(z) \to y = z$. A extenze o definici konstantního symbolu c formulí $\psi(y)$ je teorie $T' = T \cup \{c = y \leftrightarrow \psi(y)\}$.

Příklad 6.7.13. Ukážeme si dva příklady:

- Libovolnou teorii v jazyce aritmetiky můžeme rozšířit o definici konstantního symbolu 1 formulí $\psi(y)$ tvaru $y = S(0)$, přidáme tedy axiom $1 = y \ \leftrightarrow \ y = S(0)$.
- Uvažme teorii těles a nový symbol $\frac{1}{2}$, definovaný formulí $y \cdot (1 + 1) = 1$, tj. přidáním axiomu:
$$\frac{1}{2} = y \ \leftrightarrow \ y \cdot (1 + 1) = 1$$
Zde nejde o korektní extenzi o definici, neboť neplatí axiom existence. Ve dvouprvkovém tělese $\mathbb{Z}_2$ (a v každém tělese *charakteristiky 2*) nemá rovnice $y \cdot (1+1) = 1$ řešení, neboť $1 + 1 = 0$.

  Pokud ale vezmeme teorii těles charakteristiky různé od 2, tj. přidáme-li k teorii těles axiom $\neg(1 + 1 = 0)$, potom už půjde o korektní extenzi o definici. Například v tělese $\mathbb{Z}_3$ máme $\frac{1}{2}^{\mathbb{Z}_3} = 2$.

Extenze o definice

Máme-li L-teorii T a L'-teorii T', potom řekneme, že T' je *extenzí T o definice*, pokud vznikla z T postupnou extenzí o definice relačních a funkčních (příp. konstantních) symbolů. Vlastnosti, které jsme dokázali o extenzích o jeden symbol (ať už relační nebo funkční), se snadno rozšíří indukcí na více symbolů:

Důsledek 6.7.14. *Je-li T' extenze teorie T o definice, potom platí:*

- *Každý model teorie T lze jednoznačně expandovat na model T'.*
- *T' je konzervativní extenze T.*
- *Pro každou L'-formuli φ' existuje L-formule φ taková, že $T' \models \varphi' \leftrightarrow \varphi$.*

Na závěr ještě jeden příklad, na kterém si ukážeme i rozvádění definic:

Příklad 6.7.15. V teorii $T = \{(\exists y)(x + y = 0), (x + y = 0) \wedge (x + z = 0) \to y = z\}$ jazyka $L = \langle +, 0, \leq \rangle$ s rovností lze zavést $<$ a unární funkční symbol $-$ přidáním axiomů:

$$\begin{aligned} -x = y \ &\leftrightarrow \ x + y = 0 \\ x < y \ &\leftrightarrow \ x \leq y \wedge \neg(x = y) \end{aligned}$$

Formule $-x < y$ (v jazyce $L' = \langle +, -, 0, \leq, < \rangle$ s rovností) je v této extenzi o definice ekvivalentní následující formuli:

$$(\exists z)((z \leq y \wedge \neg(z = y)) \wedge x + z = 0)$$

6.8 Definovatelnost ve struktuře

Formuli s jednou volnou proměnnou x můžeme chápat jako *vlastnost* prvků. V dané struktuře taková formule *definuje* množinu prvků, které tuto vlastnost splňují, tj. takových, že formule platí při ohodnocení e, ve kterém $e(x) = a$. Máme-li formuli se dvěma volnými proměnnými, definuje binární relaci, atp. Nyní tento koncept formalizujeme. Připomeňme, že zápis $\varphi(x_1, \ldots, x_n)$ znamená, že $x_1, \ldots, x_n$ jsou právě všechny volné proměnné formule φ.

Definice 6.8.1 (Definovatelné množiny)**.** Mějme formuli $\varphi(x_1, \ldots, x_n)$ a strukturu $\mathcal{A}$ v témž jazyce. *Množina definovaná formulí* $\varphi(x_1, \ldots, x_n)$ *ve struktuře* $\mathcal{A}$, značíme $\varphi^{\mathcal{A}}(x_1, \ldots, x_n)$, je:

$$\varphi^{\mathcal{A}}(x_1, \ldots, x_n) = \{(a_1, \ldots, a_n) \in A^n \mid \mathcal{A} \models \varphi[e(x_1/a_1, \ldots, x_n/a_n)]\}$$

Zkráceně totéž zapíšeme také jako $\varphi^{\mathcal{A}}(\bar{x}) = \{\bar{a} \in A^n \mid \mathcal{A} \models \varphi[e(\bar{x}/\bar{a})]\}$.

Příklad 6.8.2. Uveďme několik příkladů:

- Formule $\neg(\exists y)E(x, y)$ definuje množinu všech *izolovaných* vrcholů v daném grafu.
- Uvažme těleso reálných čísel $\underline{\mathbb{R}}$. Formule $(\exists y)(y \cdot y = x) \wedge \neg(x = 0)$ definuje množinu všech kladných reálných čísel.
- Formule $x \leq y \wedge \neg(x = y)$ definuje v dané uspořádané množině $\langle S, \leq^S \rangle$ relaci *ostrého uspořádání* $<^S$.

Často se také hodí mluvit o vlastnostech prvků relativně k jiným prvkům dané struktury. To nelze vyjádřit čistě syntakticky, ale můžeme za některé z volných proměnných dosadit prvky struktury jako *parametry*. Zápisem $\varphi(\bar{x}, \bar{y})$ myslíme, že formule φ má volné proměnné $x_1, \ldots, x_n, y_1, \ldots, y_k$ (pro nějaká n, k).

Definice 6.8.3. Mějme formuli $\varphi(\bar{x}, \bar{y})$, kde $|\bar{x}| = n$ a $|\bar{y}| = k$, strukturu $\mathcal{A}$ v témž jazyce, a k-tici prvků $\bar{b} \in A^k$. *Množina definovaná formulí* $\varphi(\bar{x}, \bar{y})$ *s parametry* $\bar{b}$ *ve struktuře* $\mathcal{A}$, značíme $\varphi^{\mathcal{A},\bar{b}}(\bar{x}, \bar{y})$, je:

$$\varphi^{\mathcal{A},\bar{b}}(\bar{x}, \bar{y}) = \{\bar{a} \in A^n \mid \mathcal{A} \models \varphi[e(\bar{x}/\bar{a}, \bar{y}/\bar{b})]\}$$

Pro strukturu $\mathcal{A}$ a podmnožinu $B \subseteq A$ označíme $\mathrm{Df}^n(\mathcal{A}, B)$ množinu všech množin definovatelných ve struktuře $\mathcal{A}$ s parametry pocházejícími z B.

Příklad 6.8.4. Pro $\varphi(x, y) = E(x, y)$ je $\varphi^{\mathcal{G},v}(x, y)$ množina všech sousedů vrcholu v.

Pozorování 6.8.5. *Množina* $\mathrm{Df}^n(\mathcal{A}, B)$ *je uzavřená na doplněk, průnik, sjednocení, a obsahuje* $\emptyset$ *a* A^n*. Jde tedy o podalgebru potenční algebry* $\mathcal{P}(A^n)$*.*

6.8.1 Databázové dotazy

Definovatelnost nachází přirozenou aplikaci v relačních databázích, např. ve známém dotazovacím jazyce SQL. *Relační databáze* sestává z jedné nebo více *tabulek*, někdy se jim říká *relace*, řádky jedné tabulky jsou *záznamy (records)*, nebo také *tice (tuples)*. Jde tedy v principu o strukturu v čistě relačním jazyce. Představme si databázi obsahující dvě tabulky, Program a Movies, znázorněné na Obrázku 6.3.

SQL dotaz ve své nejjednodušší formě (pomineme-li např. *agregační funkce*) je v podstatě formule, a výsledkem dotazu je množina definovaná touto formulí (s parametry). Například, kdy a kde můžeme vidět film s Tomem Hanksem?

cinema	title	time
Atlas	Forrest Gump	20:00
Lucerna	Forrest Gump	21:00
Lucerna	Philadelphia	18:30
⋮	⋮	⋮

title	director	actor
Forrest Gump	R. Zemeckis	T. Hanks
Philadelphia	J. Demme	T. Hanks
Batman Returns	T. Burton	M. Keaton
⋮	⋮	⋮

Obrázek 6.3: Tabulky Program a Movies

select Program.cinema, Program.time **from** Program, Movies **where** Program.title = Movies.title **and** Movies.actor = 'T. Hanks'

Výsledkem bude množina $\varphi^{\text{Database,'T. Hanks'}}(x_{\text{cinema}}, x_{\text{time}}, y_{\text{actor}})$ definovaná ve struktuře Database $= \langle D, \text{Program}, \text{Movies}\rangle$, kde $D = \{\text{'Atlas'}, \text{'Lucerna'}, \ldots, \text{'M. Keaton'}\}$, s parametrem 'T. Hanks' následující formulí $\varphi(x_{\text{cinema}}, x_{\text{time}}, y_{\text{actor}})$:

$$(\exists y_{\text{title}})(\exists y_{\text{director}})(\text{Program}(x_{\text{cinema}}, y_{\text{title}}, x_{\text{time}}) \wedge \text{Movies}(y_{\text{title}}, y_{\text{director}}, y_{\text{actor}}))$$

6.9 Vztah výrokové a predikátové logiky

Nyní si ukážeme, jak lze výrokovou logiku 'simulovat' v logice predikátové, a to v teorii Booleových algeber. Nejprve představíme axiomy této teorie:

Definice 6.9.1 (Booleovy algebry)**.** *Teorie Booleových algeber* je teorie jazyka $L = \langle -, \wedge, \vee, \bot, \top\rangle$ s rovností sestávající z následujících axiomů:[27]

- *asociativita* $\wedge$ a $\vee$:
$$x \wedge (y \wedge z) = (x \wedge y) \wedge z$$
$$x \vee (y \vee z) = (x \vee y) \vee z$$
- *komutativita* $\wedge$ a $\vee$:
$$x \wedge y = y \wedge x$$
$$x \vee y = y \vee x$$
- *distributivita* $\wedge$ vůči $\vee$ a $\vee$ vůči $\wedge$:
$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$
$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$
- *absorpce*:
$$x \wedge (x \vee y) = x$$
$$x \vee (x \wedge y) = x$$
- *komplementace*:
$$x \wedge (-x) = \bot$$
$$x \vee (-x) = \top$$
- *netrivialita*:
$$\neg(\bot = \top)$$

Nejmenším modelem je *2-prvková Booleova algebra* $\langle\{0,1\}, f_\neg, f_\wedge, f_\vee, 0, 1\rangle$. Konečné Booleovy algebry jsou (až na *izomorfismus*) právě $\langle\{0,1\}^n, f^n_\neg, f^n_\wedge, f^n_\vee, (0,\ldots,0), (1,\ldots,1)\rangle$, kde f^n znamená, že funkci f aplikujeme po složkách.[28]

[27] Všimněte si *duality*: záměnou $\wedge$ s $\vee$ a $\bot$ s $\top$ získáme tytéž axiomy.

[28] Tyto Booleovy algebry jsou izomorfní *potenčním algebrám* $\mathcal{P}(\{1,\ldots,n\})$, izomorfismus je daný bijekcí mezi podmnožinami a jejich charakteristickými vektory.

Výroky tedy můžeme chápat jako *Booleovské termy* (a konstanty $\bot, \top$ představují pravdu a lež), pravdivostní hodnota výroku při daném ohodnocení prvovýroků je potom dána hodnotou odpovídajícího termu v 2-prvkové Booleově algebře. Kromě toho, *algebra výroků* daného výrokového jazyka nebo teorie je Booleovou algebrou (to platí i pro nekonečné jazyky).

Na druhou stranu, máme-li *otevřenou* formuli φ (bez rovnosti), můžeme reprezentovat atomické výroky pomocí prvovýroků, a získat tak výrok, který platí, právě když platí φ. Více o tomto směru se dozvíme v Kapitole 8 (o rezoluci v predikátové logice), kde se nejprve zbavíme kvantifikátorů pomocí tzv. *Skolemizace*.

Výrokovou logiku bychom také mohli zavést jako fragment logiky predikátové, pokud bychom povolili *nulární relace* (a nulární relační symboly v jazyce): $A^0 = \{\emptyset\}$, tedy na libovolné množině jsou právě dvě nulární relace $R^A \subseteq A^0$: $R^A = \emptyset = 0$ a $R^A = \{\emptyset\} = \{0\} = 1$. To ale dělat nebudeme.

Kapitola 7

Tablo metoda v predikátové logice

V této kapitole ukážeme, jak lze zobecnit *metodu analytického tabla* z výrokové na predikátovou logiku.[1] Metoda funguje velmi podobně, musíme si ale poradit *kvantifikátory*.

7.1 Neformální úvod

V této sekci tablo metodu neformálně představíme. K formálním definicím se vrátíme později. Začneme dvěma příklady, na kterých ilustruje, jak tablo metoda v predikátové logice funguje, a jak se vypořádá s kvantifikátory.

Příklad 7.1.1. Na Obrázku 7.1.1 jsou znázorněna dvě tabla. Jsou to tablo důkazy (v logice, tj. z prázdné teorie) *sentencí* $(\exists x)\neg P(x) \to \neg(\forall x)P(x)$ (vpravo) a $\neg(\forall x)P(x) \to (\exists x)\neg P(x)$ (vlevo) jazyka $L = \langle P \rangle$ (bez rovnosti), kde P je unární relační symbol. Symbol c_0 je *pomocný konstantní symbol*, který do jazyka při konstrukci tabla přidáváme.

Položky

Formule v položkách musí být vždy *sentence*, neboť potřebujeme, aby měly v daném modelu *pravdivostní hodnotu* (nezávisle na ohodnocení proměnných). To ale není zásadní omezení, chceme-li dokázat, že formule φ platí v teorii T, můžeme nejprve nahradit formuli φ a všechny axiomy T jejich *generálními uzávěry* (tj. univerzálně kvantifikujeme všechny volné proměnné). Získáme tak *uzavřenou* teorii T' a sentenci φ' a platí: $T' \models \varphi'$ právě když $T \models \varphi$.

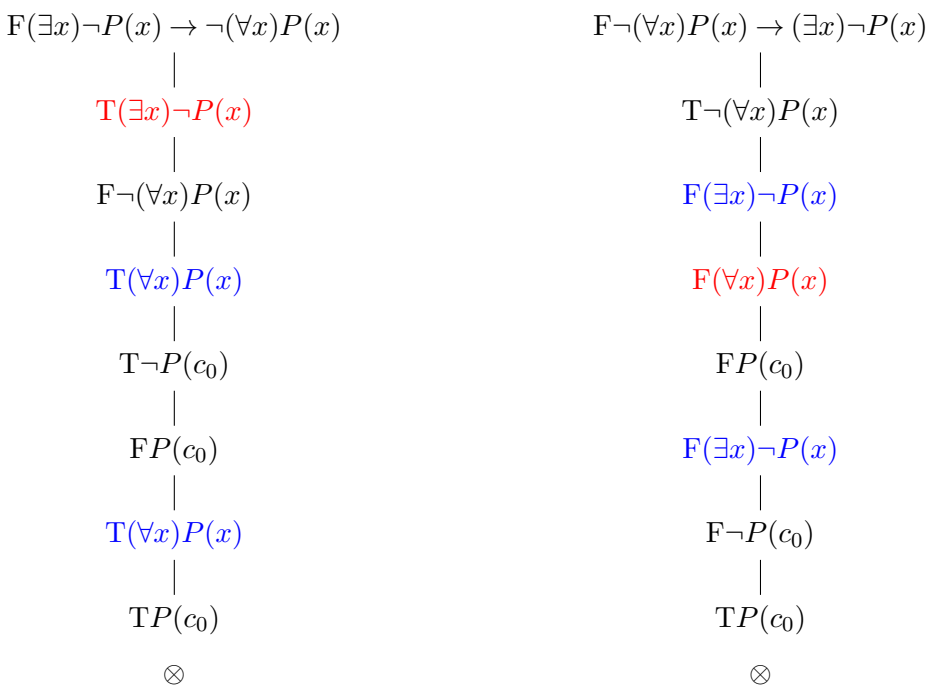
Kvantifikátory

Redukce položek funguje stejně, použijeme tatáž atomická tabla pro logické spojky (viz Tabulka 4.1, kde místo výroků jsou φ, ψ sentence). Musíme ale přidat 4 nová atomická tabla pro T/F a univerzální/existenční kvantifikátor. Tyto položky dělíme na dva typy:

- typ "*svědek*": položky tvaru $\mathrm{T}(\exists x)\varphi(x)$ a $\mathrm{F}(\forall x)\varphi(x)$
- typ "*všichni*": položky tvaru $\mathrm{T}(\forall x)\varphi(x)$ a $\mathrm{F}(\exists x)\varphi(x)$

Příklady vidíme v tablech na Obrázku 7.1.1 ('svědci' jsou červeně, 'všichni' modře).

[1] Na tomto místě je dobré připomenout si tablo metodu ve výrokové logice, viz Kapitola 4.



Obrázek 7.1: Příklady tabel. Položky typu 'svědek' jsou znázorněny červeně, položky typu 'všichni' modře.

Kvantifikátor nemůžeme pouze odstranit, neboť výsledná formule $\varphi(x)$ by nebyla sentencí. Místo toho současně s odstraněním kvantifikátoru *substituujeme* za x nějaký *konstantní term*, v nové položce tedy bude *sentence* $\varphi(x/t)$. Jaký konstantní term t substituujeme záleží na tom, zda jde o položku typu "svědek" nebo "všichni".

Pomocné konstantní symboly

Jazyk L teorie T, ve které dokazujeme, rozšíříme o spočetně mnoho *nových (pomocných) konstantních symbolů* $C = \{c_0, c_1, c_2, \dots\}$ (ale budeme psát i $c, d, \dots$), výsledný rozšířený jazyk označíme L_C. Konstantní termy v jazyce L_C tedy existují, i pokud původní jazyk L nemá žádné konstanty. A vždy při konstrukci tabla máme k dispozici nějaký *nový*, dosud *nepoužitý* (ani v teorii, ani v konstruovaném tablu) pomocný konstantní symbol $c \in C$.

Svědci

Při redukci položky typu "svědek" substituujeme za proměnnou jeden z těchto nových, pomocných symbolů, a to takový, který *dosud nebyl na dané větvi použit*. V případě položky $\mathrm{T}(\exists x)\varphi(x)$ tedy máme $\mathrm{T}\varphi(x/c)$. Tento konstantní symbol c bude hrát roli (nějakého) prvku, který danou formuli splňuje (resp. vyvrací, jde-li o položku tvaru $\mathrm{F}(\forall x)\varphi(x)$). Srovnej s Větou o konstantách (Věta 6.6.15). Je důležité, že symbol c dosud nebyl na větvi ani v teorii nijak použit. Typicky ale poté použijeme položky typu "všichni", abychom se dozvěděli, co musí *o tomto svědku platit*.

Na Obrázku 7.1.1 vidíme příklad: položka $\mathrm{T}(\exists x)\neg P(x)$ v levém tablu je redukovaná, její redukcí vznikla položka $\mathrm{T}\neg P(c_0)$; $c_0 \in C$ je pomocný symbol, na větvi se dosud nevyskytoval

(a je první takový). Podobně pro položku $\mathrm{F}(\forall x)P(x)$ a $\mathrm{F}P(c_0)$ v pravém tablu.

Všichni

Při redukci položky typu "všichni" substituujeme za proměnnou x libovolný *konstantní term* t rozšířeného jazyka L_C. Z položky tvaru $\mathrm{T}(\forall x)\varphi(x)$ tedy získáme položku $\mathrm{T}\varphi(x/t)$.

Aby byla bezesporná větev *dokončená*, budou na ní ale muset být položky $\mathrm{T}\varphi(x/t)$ pro *všechny* konstantní L_C-termy t. (Musíme 'použít' vše, co položka $\mathrm{T}(\forall x)\varphi(x)$ 'říká'.) A stejně pro položku tvary $\mathrm{F}(\exists x)\varphi(x)$.

Ve výrokové logice jsme používali konvenci, že při připojování atomických tabel vynecháváme jejich kořeny (jinak bychom opakovali na větvi tutéž položku dvakrát). V predikátové logice použijeme stejnou konvenci, ale *s výjimkou položek typu 'všichni'*. U těch zapíšeme i kořen připojovaného atomického tabla. Proč to děláme? Abychom si připomněli, že s touto položkou ještě nejsme hotovi, že musíme připojit atomická tabla s jinými konstantními termy.

Na Obrázku 7.1.1 v levém tablu *není* položka $\mathrm{T}(\forall x)P(x)$ *redukovaná*. Její *první výskyt* (4. vrchol shora) jsme zredukovali, substituujeme term $t = c_0$, máme tedy $\varphi(x/t) = P(c_0)$. Připojili jsme atomické tablo v sestávající z téže položky v kořeni $\mathrm{T}(\forall x)P(x)$, kterou do tabla *zapíšeme*, a z položky $\mathrm{T}P(c_0)$ pod ní. Zatímco *první výskyt* položky $\mathrm{T}(\forall x)P(x)$ je tímto redukovaný, *druhý výskyt* (7. vrchol shora) redukovaný není. Podobně pro položku $\mathrm{F}(\exists x)\neg P(x)$ v pravém tablu.

Tento poněkud technický přístup k definici *redukovanosti* (výskytů) položek typu 'všichni' se nám bude hodit v definici *systematického tabla*.

Jazyk

Nadále budeme předpokládat, že jazyk L je *spočetný*.[2] Z toho plyne, že každá L-teorie T má jen spočetně mnoho axiomů, a také že konstantních termů v jazyce L_C je jen spočetně mnoho. Toto omezení potřebujeme, neboť každé, i nekonečné tablo má jen spočetně mnoho položek, a musíme být schopni použít všechny axiomy dané teorie, a substituovat všechny konstantní termy jazyka L_C.

Nejprve také budeme předpokládat, že jde o jazyk *bez rovnosti*, což je jednodušší. Problémem je, že *tablo* je čistě syntaktický objekt, ale *rovnost* má speciální sémantický význam, totiž musí být v každém modelu interpretována relací identity. Jak adaptovat metodu pro jazyky s rovností si ukážeme později.

7.2 Formální definice

V této sekci definujeme všechny pojmy potřebné pro tablo metodu pro jazyky bez rovnosti. K jazykům s rovností se vrátíme v Sekci 7.3.

Buď L *spočetný* jazyk bez rovnosti. Označme jako L_C rozšíření jazyka L o spočetně mnoho nových *pomocných* konstantních symbolů $C = \{c_i \mid i \in \mathbb{N}\}$. Zvolme nějaké očíslování konstantních termů jazyka L_C, označme tyto termy $\{t_i \mid i \in \mathbb{N}\}$.

Mějme nějakou L-teorii T a L-sentenci φ.

[2] Z hlediska výpočetní logiky to není velké omezení.

7.2.1 Atomická tabla

Položka je nápis $\mathrm{T}\varphi$ nebo $\mathrm{F}\varphi$, kde φ je nějaká L_C-sentence. Položky tvaru $\mathrm{T}(\exists x)\varphi(x)$ a $\mathrm{F}(\forall x)\varphi(x)$ jsou *typu 'svědek'*, položky tvaru $\mathrm{T}(\forall x)\varphi(x)$ a $\mathrm{F}(\exists x)\varphi(x)$ jsou *typu 'všichni'*

Atomická tabla jsou položkami označkované stromy znázorněné v Tabulkách 7.1 a 7.2.

	$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$
True	$\mathrm{T}\neg\varphi$ | $\mathrm{F}\varphi$	$\mathrm{T}\varphi\wedge\psi$ | $\mathrm{T}\varphi$ | $\mathrm{T}\psi$	$\mathrm{T}\varphi\vee\psi$ / \\ $\mathrm{T}\varphi$ $\mathrm{T}\psi$	$\mathrm{T}\varphi\rightarrow\psi$ / \\ $\mathrm{F}\varphi$ $\mathrm{T}\psi$	$\mathrm{T}\varphi\leftrightarrow\psi$ / \\ $\mathrm{T}\varphi$ $\mathrm{F}\varphi$ | | $\mathrm{T}\psi$ $\mathrm{F}\psi$
False	$\mathrm{F}\neg\varphi$ | $\mathrm{T}\varphi$	$\mathrm{F}\varphi\wedge\psi$ / \\ $\mathrm{F}\varphi$ $\mathrm{F}\psi$	$\mathrm{F}\varphi\vee\psi$ | $\mathrm{F}\varphi$ | $\mathrm{F}\psi$	$\mathrm{F}\varphi\rightarrow\psi$ | $\mathrm{T}\varphi$ | $\mathrm{F}\psi$	$\mathrm{F}\varphi\leftrightarrow\psi$ / \\ $\mathrm{T}\varphi$ $\mathrm{F}\varphi$ | | $\mathrm{F}\psi$ $\mathrm{T}\psi$

Tabulka 7.1: Atomická tabla pro logické spojky; φ a ψ jsou libovolné L_C-sentence.

	$\forall$	$\exists$
True	$\mathrm{T}(\forall x)\varphi(x)$ | $\mathrm{T}\varphi(x/t_i)$	$\mathrm{T}(\exists x)\varphi(x)$ | $\mathrm{T}\varphi(x/c_i)$
False	$\mathrm{F}(\forall x)\varphi(x)$ | $\mathrm{F}\varphi(x/c_i)$	$\mathrm{F}(\exists x)\varphi(x)$ | $\mathrm{F}\varphi(x/t_i)$

Tabulka 7.2: Atomická tabla pro kvantifikátory; φ je L_C-sentence, x proměnná, t_i libovolný konstantní L_C-term, $c_i \in C$ je nový pomocný konstantní symbol (který se dosud nevyskytuje na dané větvi konstruovaného tabla).

7.2.2 Tablo důkaz

Definice v této části jsou téměř identické odpovídajícím definicím z výrokové logiky. Hlavní technický problém je jak definovat redukovanost položek typu 'všichni' na větvi tabla: chceme aby za proměnnou byly substituovány *všechny* konstantní L_C-termy t_i.

Definice 7.2.1 (Tablo)**.** *Konečné tablo z teorie T* je uspořádaný, položkami označkovaný strom zkonstruovaný aplikací konečně mnoha následujících pravidel:

- jednoprvkový strom označkovaný libovolnou položkou je tablo z teorie T,

- pro libovolnou položkou P na libovolné větvi V, můžeme na konec větve V připojit atomické tablo pro položku P, přičemž je-li P typu 'svědek', můžeme použít jen pomocný konstantní symbol $c_i \in C$, který se na větvi V dosud nevyskytuje (pro položky typu 'všichni' můžeme použít libovolný konstantní L_C-term t_i),
- na konec libovolné větve můžeme připojit položku Tα pro libovolný axiom teorie $\alpha \in T$.

Tablo z teorie T je buď konečné, nebo i *nekonečné*: v tom případě vzniklo ve spočetně mnoha krocích. Můžeme ho formálně vyjádřit jako sjednocení $\tau = \bigcup_{i \geq 0} \tau_i$, kde τ_i jsou konečná tabla z T, τ_0 je jednoprvkové tablo, a τ_{i+1} vzniklo z τ_i v jednom kroku.[3]

Tablo *pro položku* P je tablo, které má položku P v kořeni.

Připomeňme konvenci, že pokud P *není* typu 'všichni', potom kořen atomického tabla nebudeme zapisovat (neboť vrchol s položkou P už v tablu je).

Cvičení 7.1. Ukažte v jednotlivých krocích jak byla tabla z Obrázku 7.1.1 zkonstruována.

Definice 7.2.2 (Tablo důkaz)**.** *Tablo důkaz* sentence φ z teorie T je *sporné* tablo z teorie T s položkou Fφ v kořeni. Pokud existuje, je φ *(tablo) dokazatelná* z T, píšeme $T \vdash \varphi$. (Definujme také *tablo zamítnutí* jako sporné tablo s Tφ v kořeni. Pokud existuje, je φ *(tablo) zamítnutelná* z T, tj. platí $T \vdash \neg\varphi$.)

- Tablo je *sporné*, pokud je každá jeho větev sporná.
- Větev je *sporná*, pokud obsahuje položky Tψ a Fψ pro nějakou sentenci ψ, jinak je *bezesporná*.
- Tablo je *dokončené*, pokud je každá jeho větev dokončená.
- Větev je *dokončená*, pokud
  - je sporná, nebo
  - je každá položka na této větvi *redukovaná* a zároveň větev obsahuje položku Tα pro každý axiom $\alpha \in T$.
- Položka P je *redukovaná* na větvi V procházející touto položkou, pokud
  - je tvaru Tψ resp. Fψ pro *atomickou sentenci* ψ (tj. $R(t_1, \ldots, t_n)$, kde t_i jsou *konstantní* L_C-termy), nebo
  - není typu 'všichni' a vyskytuje se na V jako kořen atomického tabla[4] (tj., typicky, při konstrukci tabla již došlo k jejímu rozvoji na V), nebo
  - je typu 'všichni' a všechny její *výskyty* na V jsou na větvi V *redukované*.
- Výskyt položky P typu 'všichni' na větvi V je *i-tý*, pokud má na V právě $i-1$ předků označených touto položkou, a i-tý výskyt je *redukovaný* na V, pokud
  - položka P má $(i+1)$-ní výskyt na V, a zároveň
  - na V se vyskytuje položka T$\varphi(x/t_i)$ (je-li $P = \mathrm{T}(\forall x)\varphi(x)$) resp. F$\varphi(x/t_i)$ (je-li $P = \mathrm{F}(\exists x)\varphi(x)$), kde t_i je i-tý konstantní L_C-term.[5]

[3] Sjednocení proto, že v jednotlivých krocích přidáváme do tabla nové vrcholy, τ_i je tedy podstromem τ_{i+1}.
[4] Byť podle konvence tento kořen nezapisujeme.
[5] Tj. (typicky) už jsme za x substituovali term t_i.

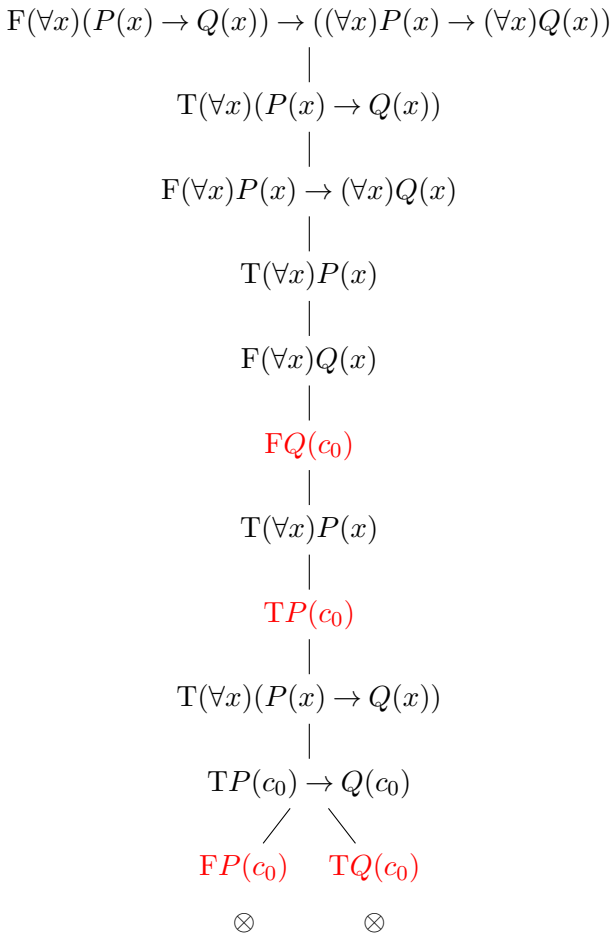
Všimněte si, že je-li položka typu ‘všichni’ na nějaké větvi redukovaná, musí mít na této větvi nekonečně mnoho výskytů, a museli jsme v nich použít při substituci všechny možnosti, tj. všechny konstantní L_C-termy.

Příklad 7.2.3. Jako příklad sestrojme tablo důkazy *v logice* (z prázdné teorie) následujících sentencí:

(a) $(\forall x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\forall x)Q(x))$, kde P, Q jsou unární relační symboly.

(b) $(\forall x)(\varphi(x) \wedge \psi(x)) \leftrightarrow ((\forall x)\varphi(x) \wedge (\forall x)\psi(x))$, kde $\varphi(x), \psi(x)$ jsou libovolné formule s jedinou volnou proměnnou x.

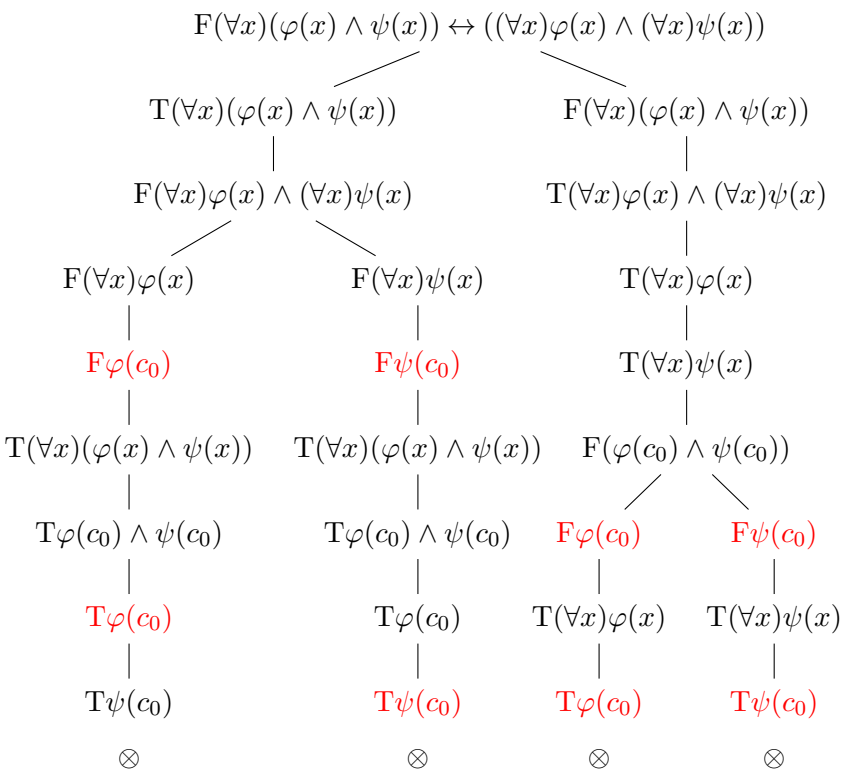
Výsledná tabla jsou na Obrázcích 7.2 a 7.3. Dvojice sporných položek jsou znázorněny červeně. Rozmyslete si, jak byla tabla po krocích zkonstruována.



Obrázek 7.2: Tablo důkaz z Příkladu 7.2.3 (a).

7.2.3 Systematické tablo a konečnost důkazů

V Sekci 4.4 jsme ukázali, že neprodlužujeme-li sporné větve (což nemusíme dělat), potom sporné tablo, speciálně tablo důkaz, bude vždy konečný. Stejný důkaz funguje i v logice predikátové.



Obrázek 7.3: Tablo důkaz z Příkladu 7.2.3 (b). Konstantu c_0 můžeme použít jako *novou* ve všech třech případech. Stačí, že se zatím nevyskytuje *na dané větvi*.

Důsledek 7.2.4 (Konečnost důkazů)**.** *Pokud $T \vdash \varphi$, potom existuje i* konečný *tablo důkaz φ z T.*

Důkaz. Stejný jako ve ve výrokové logice, viz důkaz Důsledku 4.4.5. □

Ve stejné sekci jsme si ukázali konstrukci *systematického tabla*. Tu lze také snadno adaptovat na predikátovou logiku. Musíme zajistit, abychom někdy zredukovali každou položku, použili každý axiom, a nově v predikátové logice také substituovali každý L_C term t_i za proměnnou v položkách typu 'všichni'.

Definice 7.2.5. Mějme položku R a teorii $T = \{\alpha_0, \alpha_1, \alpha_2, \dots\}$. *Systematické tablo* z teorie T pro položku R je tablo $\tau = \bigcup_{i \geq 0} \tau_i$, kde τ_0 je jednoprvkové tablo s položkou R, a pro každé $i \geq 0$:

Buď P položka v nejlevějším vrcholu v na co nejmenší úrovni tabla τ_i, která není redukovaná na nějaké bezesporné větvi procházející P (resp. jde-li o položku typu 'všichni', její *výskyt* v tomto vrcholu není redukovaný). Potom τ_i' je tablo vzniklé z τ_i připojením atomického tabla pro P na každou bezespornou větev procházející v, kde

- je-li P typu 'všichni' a má-li ve vrcholu v k-tý výskyt, potom za proměnnou substituujeme k-tý L_C-term t_k,
- je-li P typu 'svědek', potom na dané větvi V za proměnnou substituujeme $c_i \in C$ s nejmenším možným i (takovým, že na V se c_i dosud nevyskytuje).

Jinak, pokud taková položka P a vrchol v neexistují, tj. všechny položky jsou redukované, definujeme $\tau_i' = \tau_i$.

Tablo τ_{i+1} je potom tablo vzniklé z τ_i' připojením Tα_i na každou bezespornou větev τ_i', pokud $i \leq |T|$. Jinak (je-li T konečná a už jsme použili všechny axiomy) tento krok přeskočíme a definujeme $\tau_{i+1} = \tau_i'$.

Stejně jako ve výrokové logice platí, že systematické tablo je vždy dokončené, a poskytuje konečný důkaz:

Lemma 7.2.6. *Systematické tablo je dokončené.*

Důkaz. Obdobný jako důkaz ve výrokové logice (Lemma 4.4.2). Pro položky typu 'všichni' si všimněte, že k-tý výskyt redukujeme v momentě, kdy na něj při konstrukci narazíme: připojením vrcholu s $(k+1)$-ním výskytem a substitucí k-tého L_C-termu t_k. □

Důsledek 7.2.7 (Systematičnost důkazů)**.** *Pokud $T \vdash \varphi$, potom systematické tablo je (konečným) tablo důkazem φ z T.*

Důkaz. Stejný jako důkaz ve výrokové logice (Důsledek 4.4.6). □

7.3 Jazyky s rovností

Nyní si ukážeme, jak aplikovat tablo metodu na jazyky s rovností. Co je to rovnost? V matematice může v různém kontextu znamenat různé relace. Platí $1 + 0 = 0 + 1$? Mluvíme-li

o celých číslech, pak ano, ale máme-li na mysli aritmetické výrazy (nebo např. termy v jazyce těles), potom si levá a pravá strana nejsou rovny: jde o jiné výrazy. [6]

Představte si, že máme teorii T v jazyce s rovností obsahujícím konstantní symboly c_1, c_2, unární funkční symbol f a unární relační symbol P. Mějme nějaké dokončené tablo z této teorie, a v něm bezespornou větev, na kterém najdeme položku $\mathrm{T}c_1 = c_2$. Budeme chtít sestrojit *kanonický model* $\mathcal{A}$ pro tuto větev, podobně jako ve výrokové logice. Položka bude znamenat, že v kanonickém modelu platí $c_1^{\mathcal{A}} =^{\mathcal{A}} c_2^{\mathcal{A}}$, tj. $(c_1^{\mathcal{A}}, c_2^{\mathcal{A}}) \in =^{\mathcal{A}}$. To nám ale nestačí, chceme také, aby platilo také např.:

- $c_2^{\mathcal{A}} =^{\mathcal{A}} c_1^{\mathcal{A}}$,
- $f^{\mathcal{A}}(c_1^{\mathcal{A}}) =^{\mathcal{A}} f^{\mathcal{A}}(c_2^{\mathcal{A}})$,
- $c_1^{\mathcal{A}} \in P^{\mathcal{A}}$, právě když $c_2^{\mathcal{A}} \in P^{\mathcal{A}}$.

Obecně tedy chceme, aby relace $=^{\mathcal{A}}$ byla tzv. *kongruencí*,[7] tj. ekvivalencí, která se chová 'dobře' vůči funkcím a relacím struktury $\mathcal{A}$. Toho docílíme tak, že k teorii T přidáme tzv. *axiomy rovnosti*, které tyto vlastnosti vynutí, a tablo sestrojíme z výsledné teorie T^*.

V modelu $\mathcal{A}$ potom bude relace $=^{\mathcal{A}}$ kongruencí. To nám ale nestačí, chceme, aby rovnost byla *identita*, tj. aby $(a, b) \in =^{\mathcal{A}}$ platilo jedině když a a b jsou týmž prvkem univerza. Toho docílíme identifikací všech $=^{\mathcal{A}}$-ekvivalentních prvků do jediného prvku. Této konstrukci se říká *faktorstruktura* podle kongruence $=^{\mathcal{A}}$.[8] Nyní tyto pojmy formalizujeme.

Definice 7.3.1 (Kongruence)**.** Mějme ekvivalenci $\sim$ na množině A, funkci $f\colon A^n \to A$, a relaci $R \subseteq A^n$. Říkáme, že $\sim$ je

- *kongruencí pro funkci* f, pokud pro všechna $a_i, b_i \in A$ taková, že $a_i \sim b_i$ $(1 \leq i \leq n)$ platí $f(a_1, \ldots, a_n) \sim f(b_1, \ldots, b_n)$,
- *kongruencí pro relaci* R, pokud pro všechna $a_i, b_i \in A$ taková, že $a_i \sim b_i$ $(1 \leq i \leq n)$ platí $(a_1, \ldots, a_n) \in R$ právě když $(b_1, \ldots, b_n) \in R$.

Kongruence struktury $\mathcal{A}$ je ekvivalence $\sim$ na množině A, která je kongruencí pro všechny funkce a relace $\mathcal{A}$.

Definice 7.3.2 (Faktorstruktura)**.** Mějme strukturu $\mathcal{A}$ a její kongruenci $\sim$. *Faktorstruktura (podílová struktura)* $\mathcal{A}$ podle $\sim$ je struktura $\mathcal{A}/_\sim$ v témž jazyce, jejíž univerzum $A/_\sim$ je množina všech rozkladových tříd A podle $\sim$, a jejíž funkce a relace jsou definované *pomocí reprezentantů*, tj:

- $f^{\mathcal{A}/\sim}([a_1]_\sim, \ldots, [a_n]_\sim) = [f^{\mathcal{A}}(a_1, \ldots, a_n)]_\sim$, pro každý ($n$-ární) funkční symbol f, a
- $R^{\mathcal{A}/\sim}([a_1]_\sim, \ldots, [a_n]_\sim)$ právě když $R^{\mathcal{A}}(a_1, \ldots, a_n)$, pro každý ($n$-ární) relační symbol R.

Definice 7.3.3 (Axiomy rovnosti)**.** *Axiomy rovnosti* pro jazyk L s rovností jsou následující:

[6] Podobně např. $t_1 = t_2$ v Prologu neznamená, že jde o tentýž term, ale že termy t_1 a t_2 jsou *unifikovatelné*, viz kapitola o rezoluci v predikátové logice.

[7] Název pochází z kongruence modulo n, která je kongruencí v tomto smyslu na množině všech celých čísel, např. splňuje: $a + b \equiv c + d \pmod n$ kdykoliv $a \equiv c \pmod n$ a $b \equiv d \pmod n$.

[8] Stejně jako grupa $\mathbb{Z}_n$ je faktorstrukturou grupy $\mathbb{Z}$ podle $\equiv \pmod n$; např. prvek $2 \in \mathbb{Z}_n$ představuje množinu všech celých čísel, jejichž zbytek po dělení n je roven 2.

(i) $x = x$,

(ii) $x_1 = y_1 \wedge \cdots \wedge x_n = y_n \to f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$ pro každý n-ární funkční symbol f jazyka L,

(iii) $x_1 = y_1 \wedge \cdots \wedge x_n = y_n \to (R(x_1, \ldots, x_n) \to R(y_1, \ldots, y_n))$ pro každý n-ární relační symbol R jazyka L *včetně rovnosti.*

Cvičení 7.2. První z axiomů rovnosti znamená reflexivitu relace $=^{\mathcal{A}}$. Kam se poděly symetrie a tranzitivita? Ukažte, že plynou z axiomu (iii) pro symbol rovnosti $=$.

Z axiomů (*i*) a (*iii*) tedy plyne, že relace $=^{\mathcal{A}}$ je ekvivalence na A, a axiomy (*ii*) a (*iii*) vyjadřují, že $=^{\mathcal{A}}$ je kongruencí $\mathcal{A}$. V tablo metodě v případě jazyka s rovností implicitně přidáme všechny axiomy rovnosti:

Definice 7.3.4 (Tablo důkaz s rovností)**.** Je-li T teorie v jazyce L s rovností, potom označme jako T^* rozšíření teorie T o generální uzávěry[9] axiomů rovnosti pro jazyk L. *Tablo důkaz* z teorie T je *tablo důkaz* z T^*, podobně pro tablo zamítnutí (a obecně jakékoliv tablo).

Platí následující jednoduché pozorování:

Pozorování 7.3.5. *Jestliže $\mathcal{A} \models T^*$, potom platí i $\mathcal{A}/_{=^{\mathcal{A}}} \models T^*$, a ve struktuře $\mathcal{A}/_{=^{\mathcal{A}}}$ je symbol rovnosti interpretován jako identita. Na druhou stranu, v každém modelu, ve kterém je symbol rovnosti interpretován jako identita, platí axiomy rovnosti.*

Toto pozorování využijeme při konstrukci *kanonického modelu*, který budeme potřebovat v důkazu Věty o úplnosti. Nejprve ale dokážeme Větu o korektnosti.

7.4 Korektnost a úplnost

V této sekci dokážeme, že tablo metoda je i v predikátové logice korektní a úplná. Důkazy obou vět mají stejnou strukturu jako ve výrokové logice, liší se jen v implementačních detailech.

7.4.1 Věta o korektnosti

Model (struktura) $\mathcal{A}$ se *shoduje* s položkou P, pokud $P = \mathrm{T}\varphi$ a $\mathcal{A} \models \varphi$, nebo $P = \mathrm{F}\varphi$ a $\mathcal{A} \not\models \varphi$. Dále $\mathcal{A}$ se shoduje s větví V, pokud se shoduje s každou položkou na této větvi.

Ukážeme nejprve pomocné lemma analogické Lemmatu 4.5.1:

Lemma 7.4.1. *Shoduje-li se model $\mathcal{A}$ teorie T s položkou v kořeni tabla z teorie T (v jazyce L), potom lze $\mathcal{A}$ expandovat do jazyka L_C tak, že se shoduje s některou větví v tablu.*

Všimněte si, že stačí expandovat $\mathcal{A}$ o nové konstanty $c^{\mathcal{A}}$ vyskytující se na větvi V. Ostatní konstantní symboly lze interpretovat libovolně.

Důkaz. Mějme tablo $\tau = \bigcup_{i \geq 0} \tau_i$ z teorie T a model $\mathcal{A} \in \mathrm{M}_L(T)$ shodující se s kořenem τ, tedy s (jednoprvkovou) větví V_0 v (jednoprvkovém) τ_0.

Indukcí podle i najdeme posloupnost větví V_i a expanzí $\mathcal{A}_i$ modelu $\mathcal{A}$ o konstanty $c^{\mathcal{A}} \in C$ vyskytující se na V_i takových, že V_i je větev v tablu τ_i shodující se s modelem $\mathcal{A}_i$, V_{i+1} je prodloužením V_i, a $\mathcal{A}_{i+1}$ je expanzí $\mathcal{A}_i$ (mohou si být i rovny). Požadovaná větev tabla τ je

[9] Neboť v tablo metodě potřebujeme *sentence*.

potom $V = \bigcup_{i \geq 0} V_i$. Expanzi modelu $\mathcal{A}$ do jazyka L_C získáme jako 'limitu' expanzí $\mathcal{A}_i$, tj. vyskytuje-li se symbol $c \in C$ na V, vyskytuje se na nějaké z větví V_i a interpretujeme ho stejně jako v $\mathcal{A}_i$ (ostatní pomocné symboly interpretujeme libovolně).

- Pokud τ_{i+1} vzniklo z τ_i bez prodloužení větve V_i, definujeme $V_{i+1} = V_i$ a $\mathcal{A}_{i+1} = \mathcal{A}_i$.
- Pokud τ_{i+1} vzniklo z τ_i připojením položky $\mathrm{T}\alpha$ (pro nějaký axiom $\alpha \in T$) na konec větve V_i, definujeme V_{i+1} jako tuto prodlouženou větev a $\mathcal{A}_{i+1} = \mathcal{A}_i$ (nepřidali jsme žádný nový pomocný konstantní symbol). Protože $\mathcal{A}_{i+1}$ je modelem T, platí v něm axiom α, tedy shoduje se i s novou položkou $\mathrm{T}\alpha$.
- Nechť τ_{i+1} vzniklo z τ_i připojením atomického tabla pro nějakou položku P na konec větve V_i. Protože se model $\mathcal{A}_i$ shoduje s položkou P (která leží na větvi V_i), shoduje se i s kořenem připojeného atomického tabla.
  - Pokud jsme připojili atomické tablo pro logickou spojku, položíme $\mathcal{A}_{i+1} = \mathcal{A}_i$ (nepřidali jsme nový pomocný symbol). Protože $\mathcal{A}_{i+1}$ se shoduje s kořenem atomického tabla, shoduje se i s některou z jeho větví (stejně jako ve výrokové logice); definujeme V_{i+1} jako prodloužení V_i o tuto větev.
  - Je-li položka P typu 'svědek': Pokud je $P = \mathrm{T}(\exists x)\varphi(x)$, potom $\mathcal{A}_i \models (\exists x)\varphi(x)$, tedy existuje $a \in A$ takové, že $\mathcal{A}_i \models \varphi(x)[e(x/a)]$. Větev V_{i+1} definujeme jako prodloužení V_i o nově přidanou položku $\mathrm{T}\varphi(x/c)$ a model $\mathcal{A}_{i+1}$ jako expanzi $\mathcal{A}_i$ o konstantu $c^A = a$. Případ $P = \mathrm{F}(\forall x)\varphi(x)$ je obdobný.
  - Je-li položka P typu 'všichni', větev V_{i+1} definujeme jako prodloužení V_i o atomické tablo. Nově přidaná položka je $\mathrm{T}\varphi(x/t)$ nebo $\mathrm{F}\varphi(x/t)$ pro nějaký L_C-term t. Předpokládejme, že jde o první z těchto dvou možností, pro druhou je důkaz analogický. Model $\mathcal{A}_{i+1}$ definujeme jako *libovolnou* expanzi $\mathcal{A}_i$ o nové konstanty vyskytující se v t. Protože $\mathcal{A}_i \models (\forall x)\varphi(x)$, platí i $\mathcal{A}_{i+1} \models (\forall x)\varphi(x)$ a tedy i $\mathcal{A}_{i+1} \models \varphi(x/t)$; model $\mathcal{A}_{i+1}$ se tedy shoduje s větví V_i.

□

Připomeňme stručně myšlenku důkazu Věty o korektnosti: Pokud by existoval důkaz a zároveň protipříklad, protipříklad by se musel shodovat s některou větví důkazu, ty jsou ale všechny sporné. Důkaz je tedy téměř stejný jako ve výrokové logice.

Věta 7.4.2 (O korektnosti)**.** *Je-li sentence φ tablo dokazatelná z teorie T, potom je φ pravdivá v T, tj. $T \vdash \varphi \;\Rightarrow\; T \models \varphi$.*

Důkaz. Předpokládejme pro spor, že $T \not\models \varphi$, tj. existuje $\mathcal{A} \in \mathrm{M}(T)$ takový, že $\mathcal{A} \not\models \varphi$. Protože $T \vdash \varphi$, existuje sporné tablo z T s $\mathrm{F}\varphi$ v kořeni. Model $\mathcal{A}$ se shoduje s $\mathrm{F}\varphi$, tedy podle Lemmatu 7.4.1 lze expandovat do jazyka L_C tak, že se expanze shoduje s nějakou větví V. Všechny větve jsou ale sporné. □

7.4.2 Věta o úplnosti

Stejně jako ve výrokové logice ukážeme, že *bezesporná* větev v *dokončeném* tablo důkazu poskytuje protipříklad: model teorie T, který se shoduje s položkou $\mathrm{F}\varphi$ v kořeni tabla, tj. neplatí v něm φ. Takových modelů může být více, definujeme proto opět jeden konkrétní, *kanonický*.

Model musí mít nějakou doménu. Jak ji získat z tabla, což je čistě syntaktický objekt? Využijeme standardní (v matematice) trik: ze syntaktických objektů uděláme sémantické. Konkrétně, za doménu zvolíme množinu všech *konstantních termů* jazyka L_C.[10] Ty chápeme jako konečné řetězce. V následujícím výkladu budeme někdy (neformálně) místo termu t psát "t", abychom zdůraznili, že v daném místě chápeme t jako řetězec znaků, a ne např. jako termovou funkci, kterou je třeba vyhodnotit.[11]

Definice 7.4.3 (Kanonický model)**.** Mějme teorii T v jazyce $L = \langle \mathcal{F}, \mathcal{R} \rangle$ a nechť V je bezesporná větev nějakého dokončeného tabla z teorie T. Potom *kanonický model* pro V je L_C-struktura $\mathcal{A} = \langle A, \mathcal{F}^{\mathcal{A}} \cup C^{\mathcal{A}}, \mathcal{R}^{\mathcal{A}} \rangle$ definovaná následovně:

Je-li jazyk L bez rovnosti, potom:

- Doména A je množina všech konstantních L_C-termů.
- Pro každý n-ární relační symbol $R \in \mathcal{R}$ a "s_1", ..., "s_n" z A:

$$(\text{“}s_1\text{”}, \ldots, \text{“}s_n\text{”}) \in R^{\mathcal{A}} \text{ právě když na větvi } V \text{ je položka } \mathrm{T}R(s_1, \ldots, s_n)$$

- Pro každý n-ární funkční symbol $f \in \mathcal{F}$ a "s_1", ..., "s_n" z A:

$$f^{\mathcal{A}}(\text{“}s_1\text{”}, \ldots, \text{“}s_n\text{”}) = \text{“}f(s_1, \ldots, s_n)\text{”}$$

Speciálně, pro konstantní symbol c máme $c^{\mathcal{A}} = \text{“}c\text{”}$.

Funkci $f^{\mathcal{A}}$ tedy interpretujeme jako 'vytvoření' nového termu ze symbolu f a vstupních termů (řetězců).

Nechť je L jazyk s rovností. Připomeňme, že naše tablo je nyní z teorie T^*, tj. z rozšíření T o axiomy rovnosti pro L. Nejprve vytvoříme kanonický model $\mathcal{B}$ pro V jako by byl L bez rovnosti (jeho doména B je tedy množina všech konstantních L_C-termů). Dále definujeme relaci $=^B$ stejně jako pro ostatní relační symboly:

$$\text{“}s_1\text{”} =^B \text{“}s_2\text{”} \text{ právě když na větvi } V \text{ je položka } \mathrm{T}s_1 = s_2$$

Kanonický model pro V potom definujeme jako faktorstrukturu $\mathcal{A} = \mathcal{B}/_{=^B}$.

Jak plyne z diskuze v Sekci 7.3, relace $=^B$ je opravdu kongruence struktury $\mathcal{B}$, definice je tedy korektní, a relace $=^{\mathcal{A}}$ je identita na A. Platí následující jednoduché pozorování:

Pozorování 7.4.4. *Pro každou formuli φ máme $\mathcal{B} \models \varphi$ (kde symbol $=$ je interpretován jako binární relace $=^B$), právě když $\mathcal{A} = \mathcal{B}/_{=^B} \models \varphi$ (kde $=$ je interpretován jako identita).*

Všimněte si, že v jazyce bez rovnosti je kanonický model vždy spočetně nekonečný. V jazyce s rovností může ale být konečný, jak uvidíme v následujících příkladech.

Příklad 7.4.5. Nejprve si ukažme příklad kanonického modelu v jazyce bez rovnosti. Mějme teorii $T = \{(\forall x)R(f(x))\}$ v jazyce $L = \langle R, f, d \rangle$ bez rovnosti, kde R je unární relační, f unární funkční, a d konstantní symbol. Najděme protipříklad ukazující, že $T \not\models \neg R(d)$.

[10] Tj. termů zbudovaných aplikací funkčních symbolů jazyka L na konstantní symboly jazyka L (má-li nějaké) a pomocné konstantní symboly z C.

[11] Srovnejte aritmetický výraz "1+1" a 1+1=2.

Systematické tablo z T s položkou $\mathrm{F}\neg R(d)$ v kořeni není sporné, obsahuje jedinou větev V, která je bezesporná. (Sestrojte si tablo sami!) Kanonický model pro V je L_C-struktura $\mathcal{A} = \langle A, R^{\mathcal{A}}, f^{\mathcal{A}}, d^{\mathcal{A}}, c_0^{\mathcal{A}}, c_1^{\mathcal{A}}, c_2^{\mathcal{A}}, \dots \rangle$, jejíž doména je

$$A = \{\text{“}d\text{”}, \text{“}f(d)\text{”}, \text{“}f(f(d))\text{”}, \dots, \text{“}c_0\text{”}, \text{“}f(c_0)\text{”}, \text{“}f(f(c_0))\text{”}, \dots, \text{“}c_1\text{”}, \text{“}f(c_1)\text{”}, \text{“}f(f(c_1))\text{”}, \dots\}$$

a interpretace symbolů jsou následující:

- $d^{\mathcal{A}} = \text{“}d\text{”}$,
- $c_i^{\mathcal{A}} = \text{“}c_i\text{”}$ pro všechna $i \in \mathbb{N}$,
- $f^{\mathcal{A}}(\text{“}d\text{”}) = \text{“}f(d)\text{”}$, $f^{\mathcal{A}}(\text{“}f(d)\text{”}) = \text{“}f(f(d))\text{”}$, …
- $R^{\mathcal{A}} = A \backslash C = \{\text{“}d\text{”}, \text{“}f(d)\text{”}, \text{“}f(f(d))\text{”}, \dots, \text{“}f(c_0)\text{”}, \text{“}f(f(c_0))\text{”}, \dots, \text{“}f(c_1)\text{”}, \text{“}f(f(c_1))\text{”}, \dots\}$.

Redukt kanonického modelu $\mathcal{A}$ na původní jazyk L je potom $\mathcal{A}' = \langle A, R^{\mathcal{A}}, f^{\mathcal{A}}, d^{\mathcal{A}} \rangle$.

Příklad 7.4.6. Nyní příklad v jazyce s rovností: Mějme teorii $T = \{(\forall x)R(f(x)), (\forall x)(x = f(f(x)))\}$ v jazyce $L = \langle R, f, d \rangle$ s rovností. Opět najděme protipříklad ukazující, že $T \not\models \neg R(d)$.

Systematické tablo z teorie T^* (tj. z T rozšířené o axiomy rovnosti pro L) s položkou $\mathrm{F}\neg R(d)$ v kořeni obsahuje bezespornou větev V. (Sestrojte si tablo sami!) Nejprve sestrojíme kanonický model $\mathcal{B}$ pro tuto větev, jako by byl jazyk bez rovnosti:

$$\mathcal{B} = \langle B, R^{\mathcal{B}}, f^{\mathcal{B}}, d^{\mathcal{B}}, c_0^{\mathcal{B}}, c_1^{\mathcal{B}}, c_2^{\mathcal{B}}, \dots \rangle$$

kde B je množina všech konstantních L_C-termů. Relace $=^B$ je definovaná, jako by symbol ‘=’ byl ‘obyčejným’ relačním symbolem v L. Je to kongruence struktury $\mathcal{B}$, a platí pro ni, že $s_1 =^B s_2$ právě když $s_1 = f(\cdots(f(s_2))\cdots)$ nebo $s_2 = f(\cdots(f(s_1))\cdots)$ pro sudý počet aplikací f. Jako reprezentanty jednotlivých tříd tedy můžeme vybrat termy s žádným nebo jedním výskytem symbolu f:

$$B/_{=^B} = \{[\text{“}d\text{”}]_{=^B}, [\text{“}f(d)\text{”}]_{=^B}, [\text{“}c_0\text{”}]_{=^B}, [\text{“}f(c_0)\text{”}]_{=^B}, [\text{“}c_1\text{”}]_{=^B}, [\text{“}f(c_1)\text{”}]_{=^B}, \dots\}$$

Kanonický model pro větev V je potom L_C-struktura

$$\mathcal{A} = \mathcal{B}/_{=^B} = \langle A, R^{\mathcal{A}}, f^{\mathcal{A}}, d^{\mathcal{A}}, c_0^{\mathcal{A}}, c_1^{\mathcal{A}}, c_2^{\mathcal{A}}, \dots \rangle$$

kde $A = B/_{=^B}$ a interpretace symbolů jsou následující:

- $d^{\mathcal{A}} = [\text{“}d\text{”}]_{=^B}$,
- $c_i^{\mathcal{A}} = [\text{“}c_i\text{”}]_{=^B}$ pro všechna $i \in \mathbb{N}$,
- $f^{\mathcal{A}}([\text{“}d\text{”}]_{=^B}) = [\text{“}f(d)\text{”}]_{=^B}$, $f^{\mathcal{A}}([\text{“}f(d)\text{”}]_{=^B}) = [\text{“}f(f(d))\text{”}]_{=^B} = [\text{“}d\text{”}]_{=^B}$, …
- $R^{\mathcal{A}} = A = B/_{=^B}$.

Redukt kanonického modelu $\mathcal{A}$ na původní jazyk L je opět $\mathcal{A}' = \langle A, R^{\mathcal{A}}, f^{\mathcal{A}}, d^{\mathcal{A}} \rangle$.

Cvičení 7.3. (a) Sestrojte dokončené tablo s položkou $\mathrm{T}(\forall x)(\forall y)(x = y)$ v kořeni. Sestrojte kanonický model pro (jedinou, bezespornou) větev tohoto tabla.

(b) Sestrojte dokončené tablo s položkou $\mathrm{T}(\forall x)(\forall y)(\forall z)(x = y \vee x = z \vee y = z)$ v kořeni. Sestrojte kanonické modely pro několik bezesporných větví a porovnejte je.

Nyní jsme připraveni dokázat Větu o úplnosti. Použijeme opět následující pomocné lemma, jehož znění je zcela stejné, jako znění Lemmatu 4.5.4 a důkaz se liší jen v technických detailech.

Lemma 7.4.7. *Kanonický model pro (bezespornou dokončenou) větev V se shoduje s V.*

Důkaz. Nejprve uvažme jazyky bez rovnosti. Ukážeme indukcí podle struktury sentencí v položkách, že kanonický model $\mathcal{A}$ se shoduje se všemi položkami P na větvi V.

Základ indukce, tj. případ, kdy $\varphi = R(s_1, \ldots, s_n)$ je atomická sentence, je jednoduchý: Je-li na V položka $\mathrm{T}\varphi$, potom $(s_1, \ldots, s_n) \in R^{\mathcal{A}}$ plyne přímo z definice kanonického modelu, máme tedy $\mathcal{A} \models \varphi$. Je-li na V položka $\mathrm{F}\varphi$, potom na V není položka $\mathrm{T}\varphi$ (V je bezesporná), $(s_1, \ldots, s_n) \notin R^{\mathcal{A}}$, a $\mathcal{A} \not\models \varphi$

Nyní indukční krok. Rozebereme jen několik případů, ostatní se dokáží obdobně.

Pro logické spojky je důkaz zcela stejný jako ve výrokové logice, například je-li $P = \mathrm{F}\varphi \wedge \psi$, potom protože je P na V redukovaná, vyskytuje se na V položka $\mathrm{F}\varphi$ nebo položka $\mathrm{F}\psi$. Platí tedy $\mathcal{A} \not\models \varphi$ nebo $\mathcal{A} \not\models \psi$, z čehož plyne $\mathcal{A} \not\models \varphi \wedge \psi$ a $\mathcal{A}$ se shoduje s P.

Máme-li položku typu "všichni", například $P = \mathrm{T}(\forall x)\varphi(x)$ (případ $P = \mathrm{F}(\exists x)\varphi(x)$ je obdobný), potom jsou na V i položky $T\varphi(x/t)$ pro každý konstantní L_C-term, tj. pro každý prvek "t" $\in A$. Dle indukčního předpokladu je $\mathcal{A} \models \varphi(x/t)$ pro každé "t" $\in A$, tedy $\mathcal{A} \models (\forall x)\varphi(x)$.

Máme-li položku typu "svědek", například $P = \mathrm{T}(\exists x)\varphi(x)$ (případ $P = \mathrm{F}(\forall x)\varphi(x)$ je obdobný), potom je na V i položka $T\varphi(x/c)$ pro nějaké "c" $\in A$. Dle indukčního předpokladu je $\mathcal{A} \models \varphi(x/c)$, tedy i $\mathcal{A} \models (\exists x)\varphi(x)$.

Je-li jazyk s rovností, máme kanonický model $\mathcal{A} = \mathcal{B}/_{=^B}$, důkaz výše platí pro $\mathcal{B}$, a zbytek plyne z Pozorování 7.4.4. □

Cvičení 7.4. Ověřte zbývající případy v důkazu Lemmatu 7.4.7.

Důkaz Věty o úplnosti je také analogický její verzi pro výrokovou logiku:

Věta 7.4.8 (O úplnosti)**.** *Je-li sentence φ pravdivá v teorii T, potom je tablo dokazatelná z T, tj. $T \models \varphi \;\Rightarrow\; T \vdash \varphi$.*

Důkaz. Ukážeme, že libovolné *dokončené* tablo z T s položkou $\mathrm{F}\varphi$ v kořeni je nutně sporné. Důkaz provedeme sporem: kdyby takové tablo nebylo sporné, existovala by v něm bezesporná (dokončená) větev V. Uvažme kanonický model $\mathcal{A}$ pro tuto větev, a označme jako $\mathcal{A}'$ jeho redukt na jazyk L. Protože je V dokončená, obsahuje $\mathrm{T}\alpha$ pro všechny axiomy $\alpha \in T$. Model $\mathcal{A}$ se podle Lemmatu 7.4.7 shoduje se všemi položkami na V, splňuje tedy všechny axiomy a máme i $\mathcal{A}' \models T$. Protože se ale $\mathcal{A}$ shoduje i s položkou $\mathrm{F}\varphi$ v kořeni, platí i $\mathcal{A}' \not\models \varphi$, což znamená, že $\mathcal{A}' \in \mathrm{M}_L(T) \setminus \mathrm{M}_L(\varphi)$, tedy $T \not\models \varphi$, a to je spor. Tablo tedy muselo být sporné, tj. být tablo důkazem φ z T. □

7.5 Důsledky korektnosti a úplnosti

Stejně jako ve výrokové logice, Věty o korektnosti a úplnosti dohromady říkají, že *dokazatelnost* je totéž, co *platnost*. To nám umožňuje obdobně zformulovat syntaktické analogie sémantických pojmů a vlastností.

Analogií *důsledků* jsou *teorémy* teorie T:

$$\mathrm{Thm}_L(T) = \{\varphi \mid \varphi \text{ je } L\text{-sentence a } T \vdash \varphi\}$$

Důsledek 7.5.1 (Dokazatelnost = platnost)**.** *Pro libovolnou teorii T a sentence φ, ψ platí:*

- $T \vdash \varphi$ *právě když* $T \models \varphi$
- $\mathrm{Thm}_L(T) = \mathrm{Csq}_L(T)$

Platí například:

- Teorie je *sporná*, jestliže je v ní dokazatelný spor (tj. $T \vdash \bot$).
- Teorie je *kompletní*, jestliže pro každou sentenci φ je buď $T \vdash \varphi$ nebo $T \vdash \neg\varphi$ (ale ne obojí, jinak by byla sporná).
- Věta o dedukci: Pro teorii T a sentence φ, ψ platí $T, \varphi \vdash \psi$, právě když $T \vdash \varphi \to \psi$.

Na závěr této sekce si ukážeme několik aplikací Vět o úplnosti a korektnosti.

7.5.1 Löwenheim-Skolemova věta

Věta 7.5.2 (Löwenheim-Skolemova)**.** *Je-li L spočetný jazyk bez rovnosti, potom každá bezesporná L-teorie má spočetně nekonečný model.*

Důkaz. Vezměme nějaké dokončené (např. systematické) tablo z teorie T s položkou $\mathrm{F}\bot$ v kořeni. Protože T je bezesporná, není v ní dokazatelný spor, tedy tablo musí obsahovat bezespornou větev. Hledaný spočetně nekonečný model je L-redukt kanonického modelu pro tuto větev. □

K této větě se ještě vrátíme v Kapitole 9, kde si ukážeme silnější verzi zahrnující i jazyky s rovností (v nich je kanonický model spočetný, ale může být i konečný).

7.5.2 Věta o kompaktnosti

Stejně jako ve výrokové logice platí Věta o kompaktnosti, stejný je i její důkaz:

Věta 7.5.3 (O kompaktnosti)**.** *Teorie má model, právě když každá její konečná část má model.*

Důkaz. Model teorie je zřejmě modelem každé její části. Naopak, pokud T nemá model, je sporná, tedy $T \vdash \bot$. Vezměme nějaký *konečný* tablo důkaz $\bot$ z T. K jeho konstrukci stačí konečně mnoho axiomů T, ty tvoří konečnou podteorii $T' \subseteq T$, která nemá model. □

7.5.3 Nestandardní model přirozených čísel

Na závěr této sekce si ukážeme, že existuje tzv. *nestandardní model* přirozených čísel. Klíčem je Věta o kompaktnosti.

Nechť $\underline{\mathbb{N}} = \langle \mathbb{N}, S, +, \cdot, 0, \leq \rangle$ je standardní model přirozených čísel. Označme $\mathrm{Th}(\underline{\mathbb{N}})$ množinu všech sentencí *pravdivých* ve struktuře $\underline{\mathbb{N}}$ (tzv. *teorii struktury* $\underline{\mathbb{N}}$). Pro $n \in \mathbb{N}$ definujme *n-tý numerál* jako term $\underline{n} = S(S(\cdots(S(0)\cdots))$, kde S je aplikováno n-krát.

Vezměme nový konstantní symbol c a vyjádřeme, že je ostře větší než každý n-tý numerál:

$$T = \mathrm{Th}(\underline{\mathbb{N}}) \cup \{\underline{n} < c \mid n \in \mathbb{N}\}$$

Všimněte si, že každá konečná část teorie T má model. Z věty o kompaktnosti tedy plyne, že i teorie T má model. Říkáme mu *nestandardní model* (označme ho $\mathcal{A}$). Platí v něm tytéž sentence, které platí ve standardním modelu, ale zároveň obsahuje prvek $c^{\mathcal{A}}$, který je větší než každé $n \in \mathbb{N}$ (čímž zde myslíme hodnotu termu $\underline{n}$ v nestandardním modelu $\mathcal{A}$).

7.6 Hilbertovský kalkulus v predikátové logice

Na závěr kapitoly si ukážeme, jak lze adaptovat Hilbertův kalkulus, představený v Sekci 4.8, pro použití v predikátové logice. To není těžké, abychom se vypořádali s kvantifikátory, stačí přidat dvě nová schémata logických axiomů a jedno nové inferenční pravidlo. Opět si ukážeme korektnost tohoto dokazovacího systému, a jen zmíníme, že je také úplný.

Důkazy budou sestávat z libovolných formulí, nejen sentencí. Připomeňme, že Hilbertovský kalkulus používá jen spojky $\neg$ a $\to$. Budeme mít obdobné logické axiomy, jako ve výrokové logice; v případě jazyka s rovností navíc přidáme *axiomy rovnosti*.

Definice 7.6.1 (Schémata axiomů v hilbertovském kalkulu v predikátové logice)**.** Pro libovolné formule φ, ψ, χ, term t, a proměnnou x jsou následující formule logickými axiomy:

(i) $\varphi \to (\psi \to \varphi)$

(ii) $(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$

(iii) $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$

(iv) $(\forall x)\varphi \to \varphi(x/t)$, je-li t substituovatelný za x do φ

(v) $(\forall x)(\varphi \to \psi) \to (\varphi \to (\forall x)\psi)$, není-li x volná ve φ

Je-li jazyk s rovností, potom jsou logickými axiomy také *axiomy rovnosti* pro daný jazyk.

Všimněte si, že všechny logické axiomy jsou opravdu tautologie. Jako odvozovací pravidla nám poslouží *modus ponens* a dále *pravidlo generalizace*:

Definice 7.6.2 (Modus ponens)**.** *Modus ponens* říká, že pokud jsme již dokázali φ a také $\varphi \to \psi$, můžeme odvodit i formuli ψ:

$$\frac{\varphi, \varphi \to \psi}{\psi}$$

Definice 7.6.3 (Pravidlo generalizace)**.** *Pravidlo generalizace* říká, že pokud jsme dokázali φ, lze odvodit i formuli $(\forall x)\varphi$ (pro libovolnou proměnnou x):

$$\frac{\varphi}{(\forall x)\varphi}$$

Všimněte si, že obě odvozovací pravidla jsou *korektní*, tj. platí-li v nějaké teorii $T \models \varphi$ a $T \models \varphi \to \psi$, máme i $T \models \psi$, a podobně platí-li $T \models \varphi$, platí i $T \models (\forall x)\varphi$.

Stejně jako ve výrokové logice, *důkaz* bude konečná posloupnost formulí, ve které je každá nově napsaná formule buď axiomem (logickým, vč. axiomu rovnosti, nebo z teorie, ve které dokazujeme), nebo lze odvodit z předchozích pomocí jednoho z odvozovacích pravidel:

Definice 7.6.4 (Hilbertovský důkaz)**.** *Hilbertovský důkaz* formule φ z teorie T je *konečná* posloupnost formulí $\varphi_0, \ldots, \varphi_n = \varphi$, ve které pro každé $i \leq n$ platí:

- φ_i je logický axiom (včetně axiomu rovnosti, je-li jazyk s rovností), nebo
- φ_i je axiom teorie ($\varphi_i \in T$), nebo
- φ_i lze odvodit z nějakých předchozích formulí φ_j, φ_k (kde $j, k < i$) pomocí modus ponens, nebo
- φ_i lze odvodit z nějaké předchozí formule φ_j (kde $j < i$) pomocí pravidla generalizace.

Existuje-li hilbertovský důkaz, je φ *(hilbertovsky) dokazatelná*, píšeme $T \vdash_H \varphi$.

I v predikátové logice je hilbertovský kalkulus korektní a úplný dokazovací systém.

Věta 7.6.5 (O korektnosti hilbertovského kalkulu)**.** *Pro každou teorii T a formuli φ platí:*

$$T \vdash_H \varphi \;\Rightarrow\; T \models \varphi$$

Důkaz. Indukcí dle indexu i ukážeme, že každá formule φ_i z důkazu (tedy i $\varphi_n = \varphi$) platí v T.

Je-li φ_i logický axiom (včetně axiomu rovnosti), $T \models \varphi_i$ platí protože logické axiomy jsou tautologie. Je-li $\varphi_i \in T$, také jistě platí $T \models \varphi_i$. Zbytek plyne z korektnosti odvozovacích pravidel. □

Pro úplnost ještě vyslovme úplnost, důkaz ale neuvedeme.

Věta 7.6.6 (O úplnosti hilbertovského kalkulu)**.** *Pro každou teorii T a formuli φ platí:*

$$T \models \varphi \;\Rightarrow\; T \vdash_H \varphi$$

Kapitola 8

Rezoluce v predikátové logice

V této kapitole si ukážeme, jak lze adaptovat rezoluční metodu, kterou jsme představili v Kapitole 5, na predikátovou logiku. Tato kapitola, poslední v části o predikátové logice, je poměrně rozsáhlá, proto uveďme přehled její struktury:

- Začneme neformálním úvodem (Sekce 8.1).

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

- V Sekci 8.2 si ukážeme, jak pomocí *Skolemizace* odstranit kvantifikátory, abychom získali otevřené formule, které už lze převést do CNF.
- V Sekci 8.3 vysvětlíme, že rezoluční zamítnutí bychom mohli hledat 'na úrovni výrokové logiky' (tzv. *grounding*), pokud bychom nejprve za proměnné substituovali 'vhodné' konstantní termy.
- V Sekci 8.4 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 5.

- Rezoluční pravidlo, rezoluční důkaz a související pojmy jsou popsány v Sekci 8.5.
- Sekce 8.6 je věnována důkazu korektnosti a úplnosti.
- Na závěr, v Sekci 8.7, popíšeme LI-rezoluci a její aplikaci v Prologu.

8.1 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 8.1.1. 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 8.1.2. 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 8.2.

Nyní se podívejme na *rezoluční pravidlo*. To je v predikátové logice složitější. Ukážeme si opět jen několik příkladů, formální definici necháme na později (Sekce 8.5).

Příklad 8.1.3. 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 8.1.4. 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 8.1.5. 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.

8.2 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 8.2.1 (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).

8.2.1 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 8.2.2 (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 8.2.3 (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 8.2.4. *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 8.2.5. *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 8.1. Dokažte Pozorování 8.2.4 a všechna pravidla z Lemmatu 8.2.5.

Ukažme si postup na jednom příkladě:

Příklad 8.2.6. 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 8.2.3:

Důkaz Tvrzení 8.2.3. Indukcí podle struktury formule φ s využitím Lemmatu 8.2.5 a Pozorování 8.2.4. □

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 8.2.3 vyslovit také takto:

Důsledek 8.2.7. *Ke každé formuli φ existuje ekvivalentní* sentence *v PNF.*

Například v Příkladě 8.2.6 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 8.2.8. 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'.

8.2.2 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 8.2.9 (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 8.2.10. 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 8.2.11. 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 8.2.12. *Mějme L-sentenci $\varphi=(\forall x_1)\dots(\forall x_n)(\exists y)\psi$ a nechť φ' je sentence*

$$(\forall x_1)\dots(\forall x_n)\psi(y/f(x_1,\dots,x_n))$$

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 $a = (f(x_1, \ldots, x_n))^{\mathcal{A}'}[e]$, 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 $a = f^A(e(x_1), \ldots, e(x_n))$. To znamená, že expanze struktury $\mathcal{A}$ vzniklá přidáním funkce f^A je modelem φ'. □

Poznámka 8.2.13. 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 8.2.14. *Sentence φ a její skolemova varianta φ_S jsou ekvisplnitelné.*

8.2.3 Skolemova věta

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

Věta 8.2.15 (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 8.2.12 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 otevřenou teorii T''', která ekvivalentní s T'' a tedy je také konzervativní extenzí T. □

Ze sémantické charakterizace konzervativní extenze snadno plyne následující důsledek:

Důsledek 8.2.16. *Ke každé teorii můžeme pomocí skolemizace najít ekvisplnitelnou otevřenou teorii.*

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 2.3.2).

8.3 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 8.3.1 (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 8.3.2. 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.

8.3.1 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 z S) může být mnoho, i nekonečně mnoho, např. kdykoliv je v jazyce alespoň jeden funkční (nekonstantní) symbol.

Příklad 8.3.3. 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 8.4 si ukážeme efektivní postup jak hledat vhodné základní instance klauzulí, pomocí tzv. *unifikace*.

[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.

8.3.2 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 7.4.3).[6]

Definice 8.3.4 (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 8.3.5. 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 8.3.6 (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.*

[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.

Důkaz. Označme jako T_{ground} množinu všech základních instancí axiomů teorie T. Zkonstruujeme systematické[7] tablo z teorie T_{ground} s položkou $\mathrm{F}\bot$ v kořeni, ale z jazyka L, bez rozšíření o pomocné konstantní symboly na jazyk L_C.[8]

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 8.3.7. 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.

Na závěr této sekce vyslovíme dva důsledky Herbrandovy věty.

Důsledek 8.3.8. *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 teorii $T = \{\neg\varphi\}$. □

Důsledek 8.3.9. *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. □

8.4 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*).

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

[8] Protože v T_{ground} nejsou žádné kvantifikátory, pomocné symboly nikde v tablu nejsou použity.

8.4.1 Substituce

Nejprve uveďme několik příkladů 'vhodných' substitucí:

Příklad 8.4.1.

- 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 $\{\neg P(a), \neg Q(b,a)\}$, 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. Substituce budeme aplikovat na termy nebo na literály (atomické formule nebo jejich negace), označme tyto dohromady jako *výrazy*.

Definice 8.4.2 (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) E *při 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 8.4.3. 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 8.4.4. 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 8.4.5 (Skládání substitucí)**.** Mějme substituce $\sigma = \{x_1/t_1, \ldots, x_n/t_n\}$ a $\tau = \{y_1/s_1, \ldots, y_m/s_m\}$. *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 8.4.6. Jsou-li σ a τ jako v Příkladu 8.4.4, 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 8.4.7. *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, kde $y_j \notin \{x_1, \ldots, x_n\}$, 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 každou proměnnou:

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

Z toho plyne, že $(\sigma\tau)\varrho$ a $\sigma(\tau\varrho)$ jsou touž substitucí.[9] □

[9]Podrobněji: používáme zřejmou vlastnost, že pro substituci π platí $\pi = \{z_1/v_1, \ldots, z_k/v_k\}$, právě když $E\pi = v_i$ pro $E = z_i$ a $E\pi = E$ je-li E proměnná různá od všech z_i.

8.4.2 Unifikační algoritmus

Které substituce jsou tedy 'vhodné'? Takové, po jejichž provedení se dané výrazy 'stanou stejnými', tj. *unifikovanými* (viz Příklad 8.4.1).

Definice 8.4.8 (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 8.4.9. 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 8.4.10. 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 8.4.11. 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 8.4.12. 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 8.4.13. *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 $u = x$ máme $u\sigma_{i+1} = x\sigma_{i+1} = t$. 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. □

8.5 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*.

8.5.1 Rezoluční pravidlo

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

Příklad 8.5.1. 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 $\sigma = \{x/f(f(z)), y/f(z), u/f(z)\}$. Po aplikaci této unifikace získáme klauzule $C_1\sigma = \{P(f(f(z))), Q(f(f(z)),f(z))\}$ a $C_2\sigma = \{\neg P(f(z)), \neg Q(f(f(z)),f(z))\}$, z nichž odvodíme klauzuli $C = \{P(f(f(z))), \neg P(f(z))\}$. Té budeme říkat *rezolventa* původních klauzulí C_1 a C_2.

Definice 8.5.2 (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.[10] Buď σ nejobecnější unifikace S.[11] *Rezolventa* klauzulí C_1 a C_2 je následující klauzule:

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

Poznámka 8.5.3. 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 $\{P(x), P(f(x))\}$ ale není unifikovatelná, bez přejmenování proměnných by to tedy nešlo.

8.5.2 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 8.5.3.

Definice 8.5.4 (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

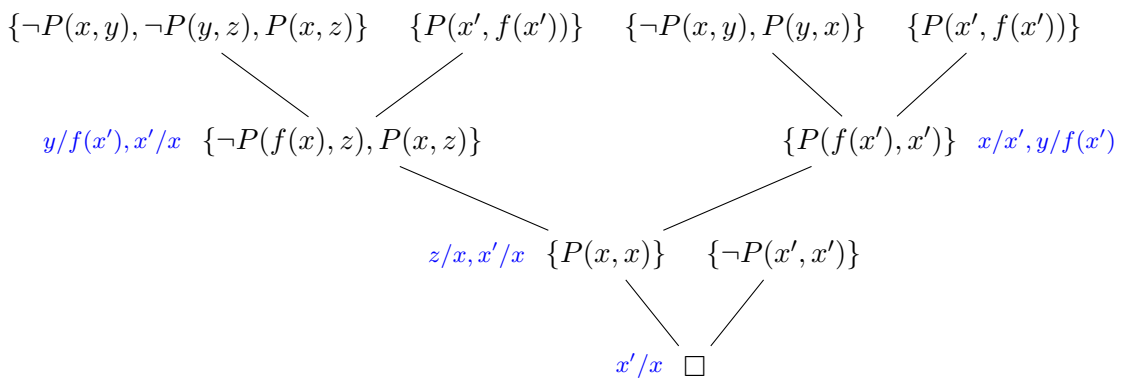
- buď $C_i = C_i'\sigma$ pro nějakou klauzuli $C_i' \in S$ a přejmenování proměnných σ, nebo
- C_i je rezolventou nějakých C_j, C_k kde $j < i$ a $k < i$.

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 8.5.5. 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.

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

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



Obrázek 8.1: Rezoluční zamítnutí formule S z Příkladu 8.5.6. U každého rezolučního kroku je zapsána použitá unifikace.

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

Příklad 8.5.6. 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 8.5.2.

8.6 Korektnost a úplnost

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

8.6.1 Věta o korektnosti

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

Tvrzení 8.6.1 (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 $\mathcal{A} \models C_2'\sigma[e]$. Tedy i $\mathcal{A} \models C[e]$. V opačném případě $\mathcal{A} \not\models A_1\sigma[e]$, musí tedy platit $\mathcal{A} \models C_1'\sigma[e]$, a opět $\mathcal{A} \models C[e]$. □

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

Věta 8.6.2 (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. □

8.6.2 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 8.6.3 (Lifting lemma)**.** *Mějme klauzule C_1 a C_2 s disjunktními množinami proměnných. Jsou-li C_1^* a C_2^* základní instance klauzulí C_1 a C_2 a je-li C^* je rezolventou C_1^* a C_2^*, potom existuje rezolventa C klauzulí C_1 a C_2 taková, že C^* je základní instancí C.*

Důkaz. Nechť $C_1^* = C_1\tau_1$ a $C_2^* = C_2\tau_2$, kde τ_1 a τ_2 jsou základní substituce, které nesdílejí žádnou proměnnou. Najdeme rezolventu C takovou, že $C^* = C\tau_1\tau_2$.

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 $(\tau_1\tau_2)$ 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 8.4.13 o korektnosti Unifikačního algoritmu víme, že $(\tau_1\tau_2) = \sigma(\tau_1\tau_2)$, což využíváme ve třetí rovnosti z následujícího výpočtu. Ve čtvrté rovnosti využíváme faktu, že $C_1'\tau_1\tau_2 = C_1'\tau_1$, a $C_2'\tau_1 = C_2'$, což plyne z toho, že jde o základní substituce nesdílející žádnou proměnnou, a že $C_1'\tau_1$ a $C_2'\tau_2$ jsou základní instance:

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

□

Indukcí podle délky rezolučního důkazu snadno získáme následující důsledek:

Důsledek 8.6.4. *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 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 8.6.5 (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 8.3.9) 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 8.6.4) 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 $S \vdash_R \square$. $\square$

8.7 LI-rezoluce

V této sekci připomeneme pojmy *lineárního a linear-input důkazu*, *LI-rezoluci* a její úplnost pro Hornovské formule. Definice i znění vět jsou stejné jako ve výrokové logice (jediným rozdílem je, že v důkazech můžeme používat *varianty* klauzulí z S), důkaz lze provést převedením na výrokovou logiku opět pomocí Herbrandovy věty a Lifting lemmatu.

Definice 8.7.1 (Lineární a LI důkaz)**.** *Lineární důkaz* (rezolucí) klauzule C z formule S je konečná posloupnost

$$\begin{bmatrix} C_0 \\ B_0 \end{bmatrix}, \begin{bmatrix} C_1 \\ B_1 \end{bmatrix}, \ldots, \begin{bmatrix} C_n \\ B_n \end{bmatrix}, C_{n+1}$$

kde C_i říkáme *centrální* klauzule, C_0 je *počáteční*, $C_{n+1} = C$ je *koncová*, B_i jsou *boční* klauzule, a platí:

- C_0 je varianta klauzule z S, pro $i \leq n$ je C_{i+1} rezolventou C_i a B_i,
- B_0 je varianta klauzule z S, pro $i \leq n$ je B_i varianta klauzule z S nebo $B_i = C_j$ pro nějaké $j < i$.

Lineární zamítnutí S je lineární důkaz $\square$ z S.

LI-důkaz je lineární důkaz, ve kterém je každá boční klauzule B_i variantou klauzule z S. Pokud existuje LI-důkaz, říkáme, že je C *LI-dokazatelná* z S, a píšeme $S \vdash_{LI} C$. Pokud $S \vdash_{LI} \square$, je S *LI-zamítnutelná.*

V Poznámce 5.4.3 jsme poznamenali, že 'lineární' rezoluce (založená na lineárních důkazech) je úplná. Důkaz byl ponechán jako cvičení. Stejné tvrzení platí i v predikátové rezoluci:

Věta 8.7.2 (O úplnosti lineární rezoluce)**.** *Klauzule C má lineární důkaz z CNF formule S, právě když má rezoluční důkaz z S (tj. $S \vdash_R C$).*

Důkaz. Z lineárního důkazu snadno vyrobíme rezoluční strom. Opačná implikace plyne z Poznámky 5.4.3 a z Lifting lemmatu (jehož použití zachovává linearitu rezolučního důkazu). $\square$

8.7.1 Úplnost LI-rezoluce pro Hornovy formule

Připomeňme terminologii týkající se hornovskosti a programů: *Hornova klauzule* je klauzule obsahující nejvýše jeden pozitivní literál. *Hornova formule* je (konečná, nebo i nekonečná) množina Hornových klauzulí. *Fakt* je pozitivní jednotková (Hornova) klauzule, *pravidlo* je (Hornova) klauzule s právě jedním pozitivním a alespoň jedním negativním literálem, a *cíl* je neprázdná (Hornova) klauzule bez pozitivního literálu. Pravidlům a faktům říkáme *programové klauzule*.

Stejně jako ve výrokové logice, LI-rezoluce je úplná pro Hornovské formule:

Věta 8.7.3 (O úplnosti LI-rezoluce pro Hornovy formule)**.** *Je-li Hornova formule T splnitelná, a $T \cup \{G\}$ je nesplnitelná pro cíl G, potom $T \cup \{G\} \vdash_{LI} \square$, a to LI-zamítnutím, které začíná cílem G.*

Důkaz. Plyne z analogické věty ve výrokové logice, z Herbrandovy věty, a z Lifting lemmatu. □

8.7.2 Rezoluce v Prologu

Na závěr si ukážeme aplikaci LI-rezoluce v programovacím jazyce Prolog. *Program* v Prologu je Hornova formule obsahující pouze *programové klauzule*, tj. *pravidla* a *fakta*.

Příklad 8.7.4. Jako příklad vezměme jednoduchý program popisující rodinné vztah třech osob, popsaný v Tabulce 8.7.4. Na levé straně vidíme syntaxi Prologu, a vpravo je množinový zápis odpovídajících klauzulí; příslušnou CNF formuli označíme P.

`son(X,Y):-father(Y,X),man(X).`	$\{son(X,Y), \neg father(Y,X), \neg man(X)\}$
`son(X,Y):-mother(Y,X),man(X).`	$\{son(X,Y), \neg mother(Y,X), \neg man(X)\}$
`man(charlie).`	$\{man(charlie)\}$
`father(bob,charlie).`	$\{father(bob, charlie)\}$
`mother(alice,charlie).`	$\{mother(alice, charlie)\}$
`?-son(charlie,X).`	$\{\neg son(charlie, X)\}$

Tabulka 8.1: Ukázkový program v Prologu

Poslední řádek v tabulce není součástí programu, jde o *existenční dotaz*. Zajímá nás, zda platí v Programu P: $P \models (\exists X) son(charlie, X)$? Všimněte si, že negací dotazu získáme *cíl* $G = \{\neg son(charlie, X)\}$. Chceme tedy *zamítnout* CNF formuli $P \cup \{G\}$.

Stejně jako ve výrokové logice (Důsledek 5.4.10) platí následující jednoduchý důsledek úplnosti LI-rezoluce pro Hornovy formule.

Důsledek 8.7.5. *Pro program P a cíl $G = \{\neg A_1, \ldots, \neg A_k\}$ v proměnných $X_1, \ldots, X_n$ jsou následující podmínky ekvivalentní:*

- $P \models (\exists X_1) \ldots (\exists X_n)(A_1 \wedge \cdots \wedge A_k)$
- *$P \cup \{G\}$ má LI-zamítnutí začínající cílem G.*

Důkaz. Není těžké nahlédnout, že program P je vždy splnitelná Hornova formule. První podmínka je ekvivalentní nesplnitelnosti $P \cup \{G\}$. Ekvivalence potom plyne z úplnosti LI-rezoluce pro Hornovy formule (Věta 8.7.3). □

Je-li odpověď na dotaz kladná, chceme znát i *výstupní substituci* σ, tj. složení unifikací z jednotlivých rezolučních kroků, zúženné na proměnné v G. Platí:

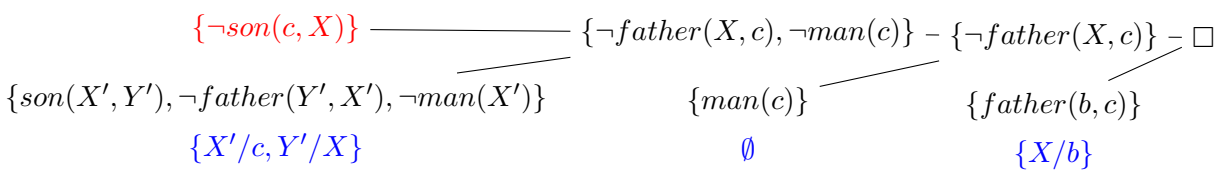
$$P \models (A_1 \wedge \cdots \wedge A_k)\sigma$$

Příklad 8.7.6. Pokračujme v Příkladu 8.7.4. Najdeme všechny výstupní substituce pro náš dotaz:

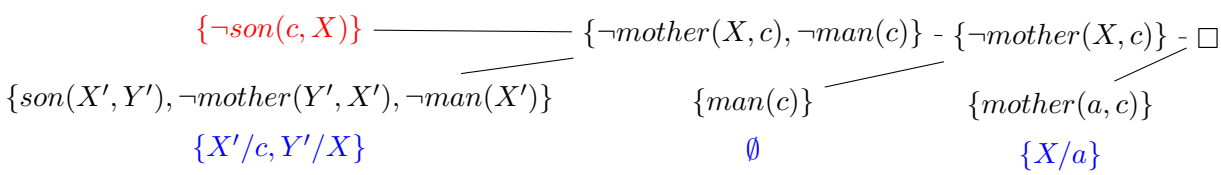
```
?-son(charlie,X).
X = bob ;
X = alice ;
No
```

Záleží na tom, které ze dvou pravidel aplikujeme na cíl. Příslušná zamítnutí jsou znázorněna níže. Výstupní substituci získáme složením substitucí z jednotlivých kroků, a zúžením na proměnnou X. (Pro nedostatek místa jsme zkrátili konstantní symboly na a, b, c.)

(a) Výstupní substituce $\sigma = \{X/b\}$:



(b) Výstupní substituce $\sigma = \{X/a\}$:



Část III

Pokročilé partie

Kapitola 9

Teorie modelů

V této kapitole se trochu vzdálíme typickým aplikacím logiky v informatice[1] a nahlédneme o úroveň abstrakce výše, do oblasti *matematické* logiky. *Teorie modelů* se snaží popsat vztah mezi obecnými vlastnostmi teorií (predikátové logiky) a tříd jejich modelů. Nevyhneme se práci s nekonečnými teoriemi a s nekonečnými strukturami. Jde jen o ukázku několika vybraných výsledků, které jsou pro nás dostupné. Ani se nepokusíme obsáhnout všechny hlavní oblasti teorie modelů, která je velmi bohatá a hluboká. Do této kapitoly jsme také přidali materiál týkající se vlastností modelů, který se nehodil jinam.

9.1 Elementární ekvivalence

Nejprve se podíváme na několik vlastností souvisejících s pojmem *elementární ekvivalence*. Připomeňme, že L-struktury $\mathcal{A}$ a $\mathcal{B}$ jsou *elementárně ekvivalentní* ($\mathcal{A} \equiv \mathcal{B}$), pokud v nich platí tytéž L-sentence.

V teorii modelů nás často zajímá, jaké vlastnosti (sentence) platí v dané, konkrétní struktuře:

Definice 9.1.1 (Teorie struktury)**.** Mějme L-strukturu $\mathcal{A}$. *Teorie struktury* $\mathcal{A}$, značíme $\mathrm{Th}(\mathcal{A})$ je množina všech L-sentencí platných v $\mathcal{A}$:

$$\mathrm{Th}(\mathcal{A}) = \{\varphi \mid \varphi \text{ je } L\text{-sentence a } \mathcal{A} \models \varphi\}$$

Příklad 9.1.2. Jako důležitý příklad vezměme *standardní model aritmetiky*, strukturu $\underline{\mathbb{N}} = \langle \mathbb{N}, S, +, \cdot, 0, \leq \rangle$. Teorii $\mathrm{Th}(\underline{\mathbb{N}})$ říkáme *aritmetika přirozených čísel*. V následující kapitole si ukážeme, že je *(algoritmicky) nerozhodnutelná*.[2]

Několik jednoduchých vlastností teorie struktury shrneme v následujícím pozorování:

Pozorování 9.1.3. *Nechť $\mathcal{A}$ je L-struktura a T je L-teorie. Potom:*

(i) Teorie $\mathrm{Th}(\mathcal{A})$ *je kompletní.*

(ii) Je-li $\mathcal{A} \in \mathrm{M}_L(T)$*, potom* $\mathrm{Th}(\mathcal{A})$ *je (kompletní) jednoduchá extenze teorie* T*.*

[1] Například použití rezoluce k řešení otázky, zda v dané konečné teorii T platí daná sentence φ.

[2] Teorie T je *(algoritmicky) rozhodnutelná*, pokud existuje algoritmus, který pro každou vstupní sentenci φ doběhne a odpoví, zda $T \models \varphi$.

(iii) Pokud $\mathcal{A} \in \mathrm{M}_L(T)$ *a* T *je kompletní, potom je* $\mathrm{Th}(\mathcal{A})$ *ekvivalentní s* T*, v tom případě* $\mathrm{Th}(\mathcal{A}) = \mathrm{Csq}_L(T)$.

Pomocí pojmu *teorie struktury* můžeme také vyjádřit elementární ekvivalenci, pro L-struktury $\mathcal{A}, \mathcal{B}$ platí:

$$\mathcal{A} \equiv \mathcal{B} \text{ právě když } \mathrm{Th}(\mathcal{A}) = \mathrm{Th}(\mathcal{B}).$$

Příklad 9.1.4. Podívejme se standardní uspořádání reálných, racionálních, a celých čísel, tj. na struktury $\langle \mathbb{R}, \leq \rangle$, $\langle \mathbb{Q}, \leq \rangle$, $\langle \mathbb{Z}, \leq \rangle$. Jak jsme již zmínili v Příkladu 6.5.3, není těžké ukázat, že $\langle \mathbb{R}, \leq \rangle \equiv \langle \mathbb{Q}, \leq \rangle$ (pomocí *hustoty* těchto uspořádání). Struktury $\langle \mathbb{Q}, \leq \rangle$ a $\langle \mathbb{Z}, \leq \rangle$ ale elementárně ekvivalentní nejsou: V $\langle \mathbb{Z}, \leq \rangle$ má každý prvek bezprostředního následníka, což v $\langle \mathbb{Q}, \leq \rangle$ neplatí. Pro následující sentenci φ tedy máme $\varphi \in \mathrm{Th}(\langle \mathbb{Z}, \leq \rangle)$ ale $\varphi \notin \mathrm{Th}(\langle \mathbb{Q}, \leq \rangle)$:

$$\varphi = (\forall x)(\exists y)(x \leq y \wedge \neg x = y \wedge (\forall z)(x \leq z \to z = x \vee y \leq z))$$

9.1.1 Kompletní jednoduché extenze

Máme-li teorii T, zajímá nás, jak vypadají její modely. Připomeňme, že:

- Teorie je *kompletní*, právě když má jediný model až na elementární ekvivalenci.[3]
- Modely teorie T až na elementární ekvivalenci jednoznačně odpovídají kompletním jednoduchým extenzím T.

Kompletní jednoduché extenze L-teorie T jsou tedy tvaru $\mathrm{Th}(\mathcal{A})$ pro $\mathcal{A} \in \mathrm{M}_L(T)$, a (jak jsme už zmínili výše) $\mathcal{A} \equiv \mathcal{B}$ právě když $\mathrm{Th}(\mathcal{A}) = \mathrm{Th}(\mathcal{B})$. Místo hledání všech modelů tedy stačí najít všechny kompletní jednoduché extenze.

Poznámka 9.1.5. Jednou z motivací, proč se zabývat kompletními jednoduchými extenzemi, je Tvrzení 10.1.6 z následující kapitoly, které říká, že pokud lze *efektivně (algoritmicky) popsat* všechny kompletní jednoduché extenze[4] *efektivně dané* teorie T,[5] potom je T *(algoritmicky) rozhodnutelná.*

Schopnost (efektivně) popsat všechny kompletní jednoduché extenze je poměrně vzácná, a vyžaduje silné předpoklady. Přesto to lze provést u mnoha důležitých teorií. Uveďme jeden příklad: *teorii hustého lineárního uspořádání (dense linear order).*

Příklad: DeLO*

Teorie *hustého lineárního uspořádání (DeLO*)* je extenze teorie uspořádání o následující axiomy:

- axiom *linearity* (někdy se mu říká také *dichotomie*):

$$x \leq y \vee y \leq x$$

- axiom *hustoty*

$$x \leq y \wedge \neg\, x = y \to (\exists z)(x \leq z \wedge z \leq y \wedge \neg\, z = x \wedge \neg\, z = y)$$

[3] Tedy všechny její modely jsou elementárně ekvivalentní.

[4] Představte si algoritmus, který pro daná vstupní i, j odpoví j-tý axiom i-té kompletní jednoduché extenze (v nějakém pevném očíslování); takový algoritmus ne vždy existuje!

[5] T může být nekonečná, ale musí existovat algoritmus, který postupně vygeneruje všechny axiomy T.

Někdy se přidává i axiom *netriviality* $(\exists x)(\exists y)(\neg\, x = y)$ zakazující jednoprvkový model. Tato teorie není kompletní, umíme ale popsat všechny její kompletní jednoduché extenze:

Tvrzení 9.1.6. *Mějme sentence $\varphi = (\exists x)(\forall y)(x \leq y)$ a $\psi = (\exists x)(\forall y)(y \leq x)$ vyjadřující existenci minimálního resp. maximálního prvku. Následující čtyři teorie jsou právě všechny (až na ekvivalenci) kompletní jednoduché extenze teorie DeLO*:*

- $\mathrm{DeLO} = \mathrm{DeLO}^* \cup \{\neg\varphi, \neg\psi\}$
- $\mathrm{DeLO}^+ = \mathrm{DeLO}^* \cup \{\neg\varphi, \psi\}$
- $\mathrm{DeLO}^- = \mathrm{DeLO}^* \cup \{\varphi, \neg\psi\}$
- $\mathrm{DeLO}^\pm = \mathrm{DeLO}^* \cup \{\varphi, \psi\}$

Stačí ukázat, že tyto čtyři teorie jsou kompletní. Potom už je zřejmé, že žádná další kompletní jednoduchá extenze DeLO* nemůže existovat. Jak vysvětlíme v Sekci 9.3, jejich kompletnost plyne z faktu, že jsou *ω-kategorické*, tj. mají jediný spočetný model až na *izomorfismus*. Viz Důsledek 9.3.5.

9.1.2 Důsledky Löwenheim-Skolemovy věty

V Sekci 7.5.1 jsme dokázali tzv. Löwenheim-Skolemovu větu, konkrétně její variantu pro jazyky bez rovnosti:

Věta (Löwenheim-Skolemova)**.** *Je-li L spočetný jazyk bez rovnosti, potom každá bezesporná L-teorie má spočetně nekonečný model.*

Tato věta má následující jednoduchý důsledek:

Důsledek 9.1.7. *Je-li L spočetný jazyk bez rovnosti, potom ke každé L-struktuře existuje elementárně ekvivalentní spočetně nekonečná struktura.*

Důkaz. Mějme L-strukturu $\mathcal{A}$. Teorie $\mathrm{Th}(\mathcal{A})$ je bezesporná (má model $\mathcal{A}$), tedy dle Löwenheim-Skolemovy věty má spočetně nekonečný model $\mathcal{B} \models \mathrm{Th}(\mathcal{A})$. To ale znamená, že $\mathcal{B} \equiv \mathcal{A}$. □

V jazyce bez rovnosti tedy nemůžeme vyjádřit například 'model má právě 42 prvků'.

V důkazu Löwenheim-Skolemovy věty jsme sestrojený model získali jako kanonický model pro bezespornou větev tabla z T pro položku $\mathrm{F}\bot$. Stejným způsobem se dokáže následující verze pro jazyky s rovností, stačí faktorizovat dle relace $=^A$:

Věta (Löwenheim-Skolemova s rovností)**.** *Je-li L spočetný jazyk s rovností, potom každá bezesporná L-teorie má spočetný model (tj. konečný, nebo spočetně nekonečný).*

I tato verze má snadný důsledek pro konkrétní struktury:

Důsledek 9.1.8. *Je-li L spočetný jazyk s rovností, potom ke každé* nekonečné *L-struktuře existuje elementárně ekvivalentní spočetně nekonečná struktura.*

Důkaz. Mějme nekonečnou L-strukturu $\mathcal{A}$. Stejně jako v důkazu Důsledku 9.1.7 (ale za použití Löwenheim-Skolemovy věty s rovností) najdeme spočetnou strukturu $\mathcal{B} \equiv \mathcal{A}$. Protože v $\mathcal{A}$ platí pro každé $n \in \mathbb{N}$ sentence vyjadřující 'existuje alespoň n prvků' (což lze pomocí rovnosti snadno zapsat), platí tato sentence i v $\mathcal{B}$, $\mathcal{B}$ tedy nemůže být konečná a musí být spočetně nekonečná. □

Tento důsledek použijeme, abychom ukázali, že existuje spočetné těleso, které je algebraicky uzavřené:

Spočetné algebraicky uzavřené těleso

Těleso $\mathcal{A}$ je *algebraicky uzavřené*, pokud každý polynom nenulového stupně v něm má kořen. Těleso reálných čísel $\mathbb{R}$ není algebraicky uzavřené, neboť $x^2 + 1$ nemá v $\mathbb{R}$ kořen, stejně tak těleso $\mathbb{Q}$ (v něm nemá kořen ani $x^2 - 2$). Těleso komplexních čísel $\mathbb{C}$ algebraicky uzavřené je, je ale nespočetné.

Algebraickou uzavřenost lze vyjádřit pomocí následujících sentencí ψ_n, pro každé $n > 0$:

$$(\forall x_{n-1}) \dots (\forall x_0)(\exists y)(y^n + x_{n-1} \cdot y^{n-1} + \dots + x_1 \cdot y + x_0) = 0$$

kde y^k je zkratka za term $y \cdot y \cdot \ \cdots \ \cdot y$ (kde $\cdot$ je aplikováno $(k-1)$-krát).

Důsledek 9.1.9. *Existuje spočetné algebraicky uzavřené těleso.*

Důkaz. Dle Důsledku 9.1.8 existuje spočetně nekonečná struktura $\mathcal{A}$ elementárně ekvivalentní tělesu $\mathbb{C}$. Protože $\mathbb{C}$ je těleso a splňuje sentence ψ_n pro všechna $n > 0$, je i $\mathcal{A}$ algebraicky uzavřené těleso. □

9.2 Izomorfismus struktur

Podívejme se blíže na pojem *izomorfismu struktur*, který zobecňuje izomorfismus grafů, vektorových prostorů, apod. Neformálně řečeno, struktury jsou *izomorfní*, pokud se liší jen pojmenováním konkrétních prvků.

Definice 9.2.1. Mějme struktury $\mathcal{A}, \mathcal{B}$ jazyka $L = \langle \mathcal{R}, \mathcal{F} \rangle$. *Izomorfismus* $\mathcal{A}$ *a* $\mathcal{B}$ (nebo '$\mathcal{A}$ *na* $\mathcal{B}$') je bijekce $h\colon A \to B$ splňující následující vlastnosti:

- Pro každý (n-ární) funkční symbol $f \in \mathcal{F}$ a pro všechna $a_i \in A$ platí:

  $$h(f^{\mathcal{A}}(a_1, \dots, a_n)) = f^{\mathcal{B}}(h(a_1), \dots, h(a_n))$$

  (Speciálně, je-li $c \in \mathcal{F}$ konstantní symbol, platí $h(c^{\mathcal{A}}) = c^{\mathcal{B}}$.)

- Pro každý (n-ární) relační symbol $R \in \mathcal{R}$ a pro všechna $a_i \in A$ platí:

  $$R^{\mathcal{A}}(a_1, \dots, a_n) \ \text{ právě když } \ R^{\mathcal{B}}(h(a_1), \dots, h(a_n))$$

Pokud existuje, říkáme, že $\mathcal{A}$ a $\mathcal{B}$ jsou *izomorfní* (nebo '$\mathcal{A}$ je *izomorfní s* $\mathcal{B}$ *via* h') a píšeme $\mathcal{A} \simeq \mathcal{B}$ (nebo $\mathcal{A} \simeq_h \mathcal{B}$). *Automorfismus* $\mathcal{A}$ je izomorfismus $\mathcal{A}$ na $\mathcal{A}$.

Všimněte si, že relace 'býti izomorfní' je ekvivalence. Ukažme si jeden příklad:

Příklad 9.2.2. Je-li $|X| = n$, je potenční algebra $\underline{\mathcal{P}(X)} = \langle \mathcal{P}(X), -, \cap, \cup, \emptyset, X \rangle$ izomorfní s Booleovou algebrou $\underline{2^n} = \langle \{0,1\}^n, -_n, \wedge_n, \vee_n, (0, \dots, 0), (1, \dots, 1) \rangle$ (kde operace aplikujeme po složkách) via $h(A) = \chi_A$, kde χ_A je charakteristický vektor podmnožiny $A \subseteq X$.

Nyní ukážeme, že izomorfismus je bijekce 'zachovávající sémantiku':

Tvrzení 9.2.3. *Mějme struktury $\mathcal{A}, \mathcal{B}$ jazyka $L = \langle \mathcal{R}, \mathcal{F} \rangle$. Bijekce $h\colon A \to B$ je izomorfismus $\mathcal{A}$ a $\mathcal{B}$, právě když platí následující:*

(i) pro každý L-term t a ohodnocení proměnných $e : \mathrm{Var} \to A$:

$$h(t^{\mathcal{A}}[e]) = t^{\mathcal{B}}[e \circ h]$$

(ii) pro každou L-formuli φ a ohodnocení proměnných $e : \mathrm{Var} \to A$:

$$\mathcal{A} \models \varphi[e] \quad \textit{právě když} \quad \mathcal{B} \models \varphi[e \circ h]$$

Důkaz. Je-li h izomorfismus, vlastnosti snadno dokážeme indukcí podle struktury termu resp. formule. Naopak, je-li h bijekce splňující (i) a (ii), dosazením $t = f(x_1, \ldots, x_n)$ resp. $\varphi = R(x_1, \ldots, x_n)$ dostáváme vlastnosti z definice izomorfismu. □

Jako okamžitý důsledek dostáváme fakt, že izomorfní struktury jsou elementárně ekvivalentní:

Důsledek 9.2.4. *Pokud $\mathcal{A} \simeq \mathcal{B}$, potom $\mathcal{A} \equiv \mathcal{B}$.*

Poznámka 9.2.5. Obrácená implikace ale obecně neplatí, například pro uspořádané množiny racionálních a reálných čísel platí $\langle \mathbb{Q}, \leq \rangle \equiv \langle \mathbb{R}, \leq \rangle$ ale $\langle \mathbb{Q}, \leq \rangle \not\simeq \langle \mathbb{R}, \leq \rangle$ neboť $\mathbb{Q}$ je spočetná množina zatímco $\mathbb{R}$ není (neexistuje tedy mezi nimi žádná bijekce).

Pro konečné modely ale platí, že izomorfismus je totéž co elementární ekvivalence, máme-li jazyk s rovností, jak dokážeme v následujícím tvrzení:

Tvrzení 9.2.6. *Je-li L jazyk s rovností a $\mathcal{A}, \mathcal{B}$ konečné L-struktury, potom platí:*

$$\mathcal{A} \simeq \mathcal{B} \quad \textit{právě když} \quad \mathcal{A} \equiv \mathcal{B}$$

Důkaz. Jednu implikaci jsme dokázali v Důsledku 9.2.4. Předpokládejme, že $\mathcal{A} \equiv \mathcal{B}$ a ukažme, že existuje izomorfismus $\mathcal{A}$ na $\mathcal{B}$. Protože je jazyk s rovností, můžeme vyjádřit sentencí, že 'existuje právě n prvků'. Z toho plyne, že $|A| = |B|$.

Označme jako $\mathcal{A}'$ expanzi $\mathcal{A}$ o jména prvků z A; jde o strukturu v jazyce $L' = L \cup \{c_a \mid a \in A\}$. Ukážeme, že $\mathcal{B}$ lze expandovat na L'-strukturu $\mathcal{B}'$ tak, že $\mathcal{A}' \equiv \mathcal{B}'$. Potom, jak lze snadno ověřit, je zobrazení $h(a) = c_a^{\mathcal{B}'}$ izomorfismem $\mathcal{A}'$ na $\mathcal{B}'$, a tedy i izomorfismem jejich L-reduktů a $\mathcal{A} \simeq \mathcal{B}$.

Stačí ukázat, že pro každé $c_a^{\mathcal{A}'} = a \in A$ existuje prvek $b \in B$ takový, že pro expanze o interpretaci konstantního symbolu c_a platí $\langle \mathcal{A}, a \rangle \equiv \langle \mathcal{B}, b \rangle$. Označme jako Ω množinu formulí $\varphi(x)$ takových, že $\langle \mathcal{A}, a \rangle \models \varphi(x/c_a)$, neboli $\mathcal{A} \models \varphi[e(x/a)]$. Protože je A konečná množina, existuje konečně mnoho formulí $\varphi_1(x), \ldots, \varphi_m(x)$ takových, že pro každou formuli $\varphi \in \Omega$ existuje i takové, že $\mathcal{A} \models \varphi \leftrightarrow \varphi_i$. Potom i $\mathcal{B} \models \varphi \leftrightarrow \varphi_i$ (neboť $\mathcal{A} \equiv \mathcal{B}$, stačí vzít generální uzávěr této formule, což je sentence).

Protože v $\mathcal{A}$ platí sentence $(\exists x) \bigwedge_{i=1}^m \varphi_i$ (je splněna díky prvku $a \in A$) a $\mathcal{B} \equiv \mathcal{A}$, máme i $\mathcal{B} \models (\exists x) \bigwedge_{i=1}^m \varphi_i$. Jinými slovy, existuje $b \in B$ takové, že $\mathcal{B} \models \bigwedge_{i=1}^m \varphi_i[e(x/b)]$. Tedy pro každou $\varphi \in \Omega$ platí $\mathcal{B} \models \varphi[e(x/b)]$, tj. $\langle \mathcal{B}, b \rangle \models \varphi(x/c_a)$, což jsme chtěli dokázat. □

Důsledek 9.2.7. *Pokud má kompletní teorie v jazyce s rovností konečný model, potom jsou všechny její modely izomorfní.*

9.2.1 Definovatelnost a automorfismy

Připomeňme si pojem definovatelné množiny, viz Sekce 6.8. Ukážeme si užitečnou vlastnost definovatelných množin: jsou uzavřené ('invariantní') na automorfismy dané struktury.

Nikoho nepřekvapí, že při automorfismu se musí izolovaný vrchol daného grafu zobrazit na izolovaný vrchol, vrchol stupně 4 na vrchol stejného stupně, nebo třeba trojice vrcholů, která tvoří trojúhelník, na trojúhelník. To nám může pomoci například při hledání automorfismů.

Tvrzení 9.2.8. *Je-li $D \subseteq A^n$ definovatelná ve struktuře $\mathcal{A}$, potom pro každý automorfismus $h \in \mathrm{Aut}(\mathcal{A})$ platí $h[D] = D$ (kde $h[D]$ značí $\{(h(\overline{a}) \mid \overline{a} \in D\}$).*

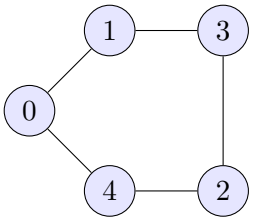
Je-li D definovatelná s parametry $\overline{b}$, platí totéž pro automorfismy identické na $\overline{b}$, tj. takové, že $h(\overline{b}) = \overline{b}$ (neboli $h(b_i) = b_i$ pro všechna i).

Důkaz. Ukážeme jen verzi s parametry. Nechť $D = \varphi^{\mathcal{A},\overline{b}}(\overline{x}, \overline{y})$. Potom pro každé $\overline{a} \in A^n$ platí následující ekvivalence:

$$\begin{aligned}
\overline{a} \in D &\Leftrightarrow \mathcal{A} \models \varphi[e(\overline{x}/\overline{a}, \overline{y}/\overline{b})] \\
&\Leftrightarrow \mathcal{A} \models \varphi[(e \circ h)(\overline{x}/\overline{a}, \overline{y}/\overline{b})] \\
&\Leftrightarrow \mathcal{A} \models \varphi[e(\overline{x}/h(\overline{a}), \overline{y}/h(\overline{b}))] \\
&\Leftrightarrow \mathcal{A} \models \varphi[e(\overline{x}/h(\overline{a}), \overline{y}/\overline{b})] \\
&\Leftrightarrow h(\overline{a}) \in D.
\end{aligned}$$

□

Příklad 9.2.9. Uvažme následující graf $\mathcal{G}$. Najděme všechny množiny definovatelné z $\mathcal{G}$ s parametrem 0, tj. množinu $\mathrm{Df}^1(\mathcal{G}, \{0\})$.



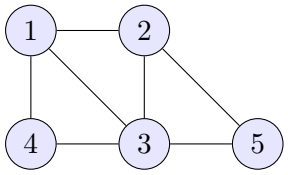
Tento graf má jediný netriviální automorfismus zachovávající vrchol 0: $h(i) = (5 - i) \bmod 5$. Jeho *orbity* jsou $\{0\}$, $\{1, 4\}$, a $\{2, 3\}$. Tyto množiny jsou definovatelné:

- $\{0\}$ je definované formulí $x = y$, tj. $(x = y)^{\mathcal{G},\{0\}} = \{0\}$,
- $\{1, 4\}$ lze definovat pomocí formule $E(x, y)$, a
- $\{2, 3\}$ formulí $\neg E(x, y) \wedge \neg x = y$.

Množina $\mathrm{Df}^1(\mathcal{G}, \{0\})$ je podalgebra potenční algebry $\mathcal{P}(V(\mathcal{G}))$, musí tedy být uzavřená na doplněk, sjednocení, průnik, a obsahovat $\emptyset$ a $V(\mathcal{G})$. Podalgebra generovaná $\{\{0\}, \{1, 4\}, \{2, 3\}\}$ už ale obsahuje všechny podmnožiny zachovávající automorfismus h. Dostáváme:

$$\mathrm{Df}^1(\mathcal{G}, \{0\}) = \{\emptyset, \{0\}, \{1, 4\}, \{2, 3\}, \{0, 1, 4\}, \{0, 2, 3\}, \{1, 4, 2, 3\}, \{0, 1, 2, 3, 4\}\}$$

Cvičení 9.1. Uvažme následující graf. Najděte všechny automorfismy. Určete, které podmnožiny jsou definovatelné, uveďte definující formule. Které binární relace jsou definovatelné?



9.3 ω-kategorické teorie

Nyní se podíváme na teorie, které mají jediný spočetně nekonečný model (až na izomorfismus), říkáme jim *ω-kategorické*.[6]

Definice 9.3.1 (Izomorfní spektrum, κ-kategoricita)**.** *Izomorfní spektrum* teorie T je počet $I(\kappa, T)$ modelů T kardinality κ až na izomorfismus, pro každou kardinalitu κ (včetně *transfinitních*). Teorie T je *κ-kategorická*, pokud $I(\kappa, T) = 1$.

Nadále nás bude zajímat jen případ $\kappa = \omega$, totiž teorie s jediným spočetně nekonečným modelem (až na izomorfismus). Jako příklad uveďme teorii hustého lineárního uspořádání bez konců:

Tvrzení 9.3.2. *Teorie DeLO je ω-kategorická.*

Důkaz. Vezměme dva spočetně nekonečné modely $\mathcal{A}, \mathcal{B}$ a očíslujme jejich prvky: $A = \{a_i \mid i \in \mathbb{N}\}$, $B = \{b_i \mid i \in \mathbb{N}\}$. Indukcí podle n lze díky hustotě nalézt posloupnost $h_0 \subseteq h_1 \subseteq h_2 \subseteq \ldots$ prostých (parciálních) funkcí z A do B, takových, že $\{a_0, \ldots, a_{n-1}\} \subseteq \operatorname{dom} h_n$, $\{b_0, \ldots, b_{n-1}\} \subseteq \operatorname{rng} h_n$,[7] a *zachovávají uspořádání*[8] Potom $\mathcal{A} \simeq \mathcal{B}$ via $h = \bigcup_{n\in\mathbb{N}} h_n$. □

Důsledek 9.3.3. *Izomorfní spektrum teorie DeLO* je následující:*

$$I(\kappa, DeLO^*) = \begin{cases} 0 & \text{pro } \kappa \in \mathbb{N}, \\ 4 & \text{pro } \kappa = \omega. \end{cases}$$

Spočetné modely až na izomorfismus jsou například:

$$\mathbb{Q} = \langle \mathbb{Q}, \leq \rangle \simeq \mathbb{Q} \restriction (0,1),\ \mathbb{Q} \restriction (0,1],\ \mathbb{Q} \restriction [0,1),\ \mathbb{Q} \restriction [0,1]$$

Důkaz. Husté uspořádání jistě nemůže být konečné. Izomorfismus musí zobrazit nejmenší prvek na nejmenší prvek, a největší na největší. □

Pojem *ω-kategoricity* lze chápat jako zeslabení pojmu *kompletnosti*. Platí následující užitečné kritérium:

Věta 9.3.4 (ω-kategorické kritérium kompletnosti)**.** *Mějme ω-kategorickou teorii T ve spočetném jazyce L. Je-li*

- *L bez rovnosti, nebo*
- *L s rovností a T nemá konečné modely,*

potom je teorie T kompletní.

Důkaz. Pro jazyk bez rovnosti víme z Důsledku 9.1.7 Löwenheim-Skolemovy věty, že každý model je elementárně ekvivalentní nějakému spočetně nekonečnému modelu. Ten je ale až na izomorfismus jediný, takže všechny modely jsou elementárně ekvivalentní, což je sémantická definice kompletnosti.

Máme-li jazyk s rovností, použijeme podobně Důsledek 9.1.8 a dostaneme, že všechny nekonečné modely jsou elementárně ekvivalentní. Mohly by existovat elementárně neekvivalentní konečné modely, to jsme ale zakázali. □

[6] Symbol ω se používá pro nejmenší nekonečné *ordinální* číslo, jinými slovy, pro množinu všech přirozených čísel.

[7] Zde dom značí *doménu* a rng značí *obor hodnot ('range')* funkce.

[8] Tj. je-li $a_i, a_j \in \operatorname{dom} h_n$, potom $a_i \leq^{\mathcal{A}} a_j$ právě když $h(a_i) \leq^{\mathcal{B}} h(a_j)$.

Důsledek 9.3.5. *Teorie* DeLO, DeLO^+, DeLO^-, *a* $\mathrm{DeLO}^\pm$ *jsou kompletní. Jsou to všechny (navzájem neekvivalentní) kompletní jednoduché extenze teorie DeLO**.

Poznámka 9.3.6. Analogické kritérium platí i pro kardinality κ větší než ω.

9.4 Axiomatizovatelnost

Na závěr této kapitoly se podíváme, za jakých okolností lze 'popsat' (*axiomatizovat*) třídu modelů respektive teorii. Zajímat nás bude také kdy si vystačíme s konečně mnoha axiomy, a kdy to lze pomocí otevřených axiomů (kterých může být i nekonečně mnoho). Srovnejte s Tvrzením 2.3.4 z výrokové logiky.

Definice 9.4.1 (Axiomatizovatelnost)**.** Mějme třídu struktur $K \subseteq \mathrm{M}_L$ v nějakém jazyce L. Říkáme, že K je

- *axiomatizovatelná*, pokud existuje L-teorie T taková, že $\mathrm{M}_L(T) = K$,
- *konečně axiomatizovatelná*, pokud je axiomatizovatelná konečnou teorií, a
- *otevřeně axiomatizovatelná*, pokud je axiomatizovatelná otevřenou teorií.

O L-teorii T' říkáme, že je *konečně* resp. *otevřeně axiomatizovatelná*, pokud to platí o třídě modelů $K = \mathrm{M}_L(T')$.

Příklad 9.4.2. Uveďme několik příkladů:

- grafy nebo částečná uspořádání jsou konečně i otevřeně axiomatizovatelné,
- tělesa jsou konečně, ale ne otevřeně axiomatizovatelná,
- nekonečné grupy jsou axiomatizovatelné, ale ne konečně,
- konečné grafy nejsou axiomatizovatelné.

Proč tomu tak je ukážeme níže.

Začněme jednoduchým faktem:

Pozorování 9.4.3. *Je-li K axiomatizovatelná, musí být uzavřená na elementární ekvivalenci.*

Z věty o kompaktnosti snadno získáme následující tvrzení, pomocí kterého lze ukázat neaxiomatizovatelnost např. konečných grafů, konečných grup, konečných těles.

Věta 9.4.4. *Pokud má teorie libovolně velké konečné modely, potom má i nekonečný model. V tom případě není třída všech jejích konečných modelů axiomatizovatelná.*

Důkaz. Je-li jazyk bez rovnosti, stačí vzít kanonický model pro některou bezespornou větev v tablu z T pro položku $\mathrm{F}\bot$ (T je bezesporná, neboť má model(y), tedy tablo není sporné).

Mějme jazyk s rovností a označme jako T' následující extenzi teorie T do jazyka rozšířeného o spočetně mnoho nových konstantních symbolů c_i:

$$T' = T \cup \{\neg c_i = c_j \mid i \neq j \in \mathbb{N}\}$$

Každá konečná část teorie T' má model: nechť k je největší takové, že symbol c_k se vyskytuje v této konečné části T'. Potom stačí vzít libovolný alespoň $(k+1)$-prvkový model T a interpretovat konstanty $c_0, \ldots, c_k$ jako navzájem různé prvky tohoto modelu.

Dle věty o kompaktnosti má potom i T' model. Ten je nutně nekonečný. Jeho redukt na původní jazyk (zapomenutí konstant $c_i^{\mathcal{A}}$) je nekonečným modelem T. □

Poznámka 9.4.5. Třída všech *nekonečných* modelů teorie ale je vždy axiomatizovatelná, máme-li jazyk s rovností: stačí k teorii přidat pro každé $n \in \mathbb{N}$ axiom vyjadřující 'existuje alespoň n prvků'.

9.4.1 Konečná axiomatizovatelnost

Ukážeme následující kritérium konečné axiomatizovatelnosti: jak třída struktur K tak i $\overline{K}$ musí být axiomatizovatelné.

Věta 9.4.6 (O konečné axiomatizovatelnosti)**.** *Mějme třídu struktur $K \subseteq \mathrm{M}_L$ a uvažme také její doplněk $\overline{K} = \mathrm{M}_L \setminus K$. Potom K je konečně axiomatizovatelná, právě když K i $\overline{K}$ jsou axiomatizovatelné.*

Důkaz. Je-li K konečně axiomatizovatelná, potom je axiomatizovatelná i konečně mnoha sentencemi $\varphi_1, \ldots, \varphi_n$ (nahradíme formule jejich generálními uzávěry). Jako axiomatizaci $\overline{K}$ stačí vzít sentenci $\psi = \neg(\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n)$. Zřejmě platí $\mathrm{M}(\psi) = \overline{K}$.

Naopak, nechť T a S jsou teorie takové, že $\mathrm{M}(T) = K$ a $\mathrm{M}(S) = \overline{K}$. Uvažme teorii $T \cup S$. Tato teorie je sporná, neboť:

$$\mathrm{M}(T \cup S) = \mathrm{M}(T) \cap \mathrm{M}(S) = K \cap \overline{K} = \emptyset$$

Podle věty o kompaktnosti[9] existují konečné podteorie $T' \subseteq T$ a $S' \subseteq S$ takové, že:

$$\emptyset = \mathrm{M}(T' \cup S') = \mathrm{M}(T') \cap \mathrm{M}(S')$$

Nyní si všimněme, že platí

$$\mathrm{M}(T) \subseteq \mathrm{M}(T') \subseteq \overline{\mathrm{M}(S')} \subseteq \overline{\mathrm{M}(S)} = \mathrm{M}(T)$$

tím jsme dokázali, že $M(T) = M(T')$, tj. teorie T' je hledanou konečnou axiomatizací K. □

Jako aplikaci si dokážeme, že tělesa charakteristiky 0 nejsou konečně axiomatizovatelná.

Příklad: tělesa charakteristiky 0

Nechť T je teorie těles. Charakteristika tělesa je nejmenší počet jedniček, které je třeba sečíst, abychom dostali nulu (v tom případě musí být charakteristika prvočíslo—dokažte si!), nebo, pokud nikdy nedostaneme sčítáním jedniček nulu, říkáme že je charakteristika 0. Trochu formálněji:

Definice 9.4.7 (Charakteristika tělesa)**.** Říkáme, že těleso $\mathcal{A} = \langle A, +, -, 0, \cdot, 1\rangle$ je

- *charakteristiky p*, je-li p nejmenší prvočíslo takové, že $\mathcal{A} \models p1 = 0$, kde $p1$ označuje term $1 + 1 + \cdots + 1$ s p jedničkami, nebo

[9] Vidíte, jak je užitečná!

- *charakteristiky 0*, pokud není charakteristiky p pro žádné prvočíslo p.

Nechť T je teorie těles. Potom třída těles charakteristiky p je konečně axiomatizována teorií $T \cup \{p1 = 0\}$. Třída těles charakteristiky 0 je axiomatizována následující (nekonečnou) teorií:

$$T' = T \cup \{\neg\, p1 = 0 \mid p \text{ je prvočíslo}\}$$

Konečná axiomatizace ale neexistuje.

Tvrzení 9.4.8. *Třída K těles charakteristiky 0 není konečně axiomatizovatelná.*

Důkaz. Díky Větě 9.4.6 stačí ukázat, že $\overline{K}$ (sestávající z těles nenulové charakteristiky a struktur, které nejsou tělesa) není axiomatizovatelná, což dokážeme sporem. Nechť existuje teorie S taková, že $\mathrm{M}(S) = \overline{K}$. Potom teorie $S' = S \cup T'$ má model, neboť každá její konečná část má model: stačí vzít těleso prvočíselné charakteristiky větší než jakékoliv p z axiomu T' tvaru $\neg\, p1 = 0$. Nechť $\mathcal{A}$ je model S'. Potom je i $\mathcal{A} \in \mathrm{M}(S) = \overline{K}$. Zároveň je ale $\mathcal{A} \in \mathrm{M}(T') = K$, což je spor. □

9.4.2 Otevřená axiomatizovatelnost

Pro otevřenou axiomatizovatelnost existuje jednoduché sémantické kritérium: třída jejích modelů musí být uzavřená na podstruktury. Platí dokonce ekvivalence, dokážeme ale jen jednu implikaci (důkaz druhé je obtížnější).

Věta 9.4.9. *Pokud je teorie T otevřeně axiomatizovatelná, potom je každá podstruktura modelu T také modelem T.*

Poznámka 9.4.10. Platí i obrácená implikace: Je-li každá podstruktura modelu T také modelem, potom je T otevřeně axiomatizovatelná. Důkaz zde ale neuvedeme.

Důkaz. Nechť T' je otevřená axiomatizace T. Mějme model $\mathcal{A} \models T'$ a podstrukturu $\mathcal{B} \subseteq \mathcal{A}$. Pro každou formuli $\varphi \in T'$ platí $\mathcal{B} \models \varphi$ (neboť φ je otevřená), tedy i $\mathcal{B} \models T'$. □

Příklad 9.4.11. Uveďme několik příkladů:

- Teorie DeLO není otevřeně axiomatizovatelná, například žádná konečná podstruktura modelu DeLO nemůže být hustá.
- Teorie těles není otevřeně axiomatizovatelná, podstruktura $\mathbb{Z} \subseteq \mathbb{Q}$ tělesa racionálních čísel není tělesem, v $\mathbb{Z}$ neexistuje inverzní prvek vůči násobení k číslu 2.
- Pro dané $n \in \mathbb{N}$ jsou nejvýše n-prvkové grupy otevřeně axiomatizovatelné(podgrupy jsou jistě také nejvýše n-prvkové). Jako otevřenou axiomatizaci lze vzít následující extenzi (otevřené) teorie grup T:

$$T \cup \{\bigvee_{1 \le i < j \le n+1} x_i = x_j\}$$

Kapitola 10

Nerozhodnutelnost a neúplnost

V této, závěrečné kapitole se budeme zabývat tím, jak lze s teoriemi pracovat algoritmicky. Zlatým hřebem budou *Gödelovy věty o neúplnosti* z roku 1931, které ukazují limity formálního přístupu, a které zastavily desetiletí trvající program formalizace matematiky. Nemáme zde dostatek prostoru k uvedení formálních definic a úplných důkazů, proto se místy budeme pohybovat na poněkud intuitivní úrovni. Zaměříme se na pochopení smyslu tvrzení a myšlenek důkazů.

Pojem *algoritmu* budeme chápat také jen intuitivně. Pokud bychom ho chtěli formalizovat, potom nejběžnější (ale zdaleka ne jedinou) volbou je koncept *Turingova stroje*.[1]

10.1 Rekurzivní axiomatizace a rozhodnutelnost

V dokazovacích systémech, kterými jsme se zabývali (tablo metoda, rezoluce, hilbertův kalkulus) jsme povolili, aby teorie T, ve které dokazujeme, byla nekonečná. Vůbec jsme se ale zatím nezabývali tím, jak je zadaná. Pokud chceme ověřit, že je daný objekt (tablo, rezoluční strom, posloupnost formulí) korektním důkazem, potřebujeme nějaký algoritmický přístup ke všem axiomům T.

Jednou z možností by bylo požadovat *enumerátor* T, tj. algoritmus, který vypisuje na výstup axiomy z T, a každý axiom někdy vypíše.[2] Potom by bylo snadné potvrdit, že je daný důkaz korektní. Pokud bychom ale dostali důkaz, který použil chybný axiom, který v T není, nikdy bychom se to nedozvěděli: nekonečně dlouho bychom čekali, zda jej enumerátor přeci jen nevypíše. Požadujeme proto silnější vlastnost, která umožňuje rozpoznat i chybné důkazy *rekurzivní axiomatizaci*:[3]

Definice 10.1.1 (Rekurzivní axiomatizace)**.** Teorie T je *rekurzivně axiomatizovaná*, pokud existuje algoritmus, který pro každou vstupní formuli φ doběhne a odpoví, zda $\varphi \in T$.

Poznámka 10.1.2. Ve skutečnosti by nám stačil enumerátor pro T, pokud by bylo garantováno, že vypisuje axiomy v lexikografickém uspořádání. To už je ekvivalentní rekurzivní axiomatizaci. (Rozmyslete si proč.)

[1] Viz přednáška NTIN090 Základy složitosti a vyčíslitelnosti.

[2] Nutným předpokladem je, aby T byla spočetná. K tomu stačí předpokládat, že jazyk je spočetný.

[3] Slovo *rekurzivní* zde neznamená běžně známou rekurzi, ale odkazuje na formalizaci algoritmu pomocí 'rekurzivních funkcí' od Gödela. Rekurzivní funkce zde znamená totéž, co vyčíslitelná nějakým Turingovým strojem, a teorii vyčíslitelnosti (*computability theory*) se někdy také říká *recursion theory*.

Zaměříme se na otázku, zda můžeme v dané teorii T 'algoritmicky rozhodovat pravdu' (tj. platnost vstupní formule). Pokud ano, říkáme, že je teorie *rozhodnutelná*. To je ale poměrně silná vlastnost, definujeme proto také *částečnou rozhodnutelnost*, která znamená, že pokud formule platí, algoritmus nám to řekne, ale pokud neplatí, nikdy se nemusíme dočkat odpovědi.

Definice 10.1.3 (Rozhodnutelnost)**.** O teorii T říkáme, že je

- *rozhodnutelná*, pokud existuje algoritmus, který pro každou vstupní formuli φ doběhne a odpoví, zda $T \models \varphi$,
- *částečně rozhodnutelná*, pokud existuje algoritmus, který pro každou vstupní formuli:
  - pokud $T \models \varphi$, doběhne a odpoví "ano",
  - pokud $T \not\models \varphi$, buď nedoběhne, nebo doběhne a odpoví "ne".

Můžeme jako obvykle předpokládat, že φ v definici je sentence. Ukážeme si jednoduché tvrzení:

Tvrzení 10.1.4. *Nechť T je rekurzivně axiomatizovaná. Potom:*

(i) T je částečně rozhodnutelná,

(ii) je-li T navíc kompletní, potom je rozhodnutelná.

Důkaz. Algoritmem ukazujícím částečnou rozhodnutelnost je konstrukce systematického tabla pro Fφ.[4] Pokud φ v T platí, konstrukce skončí v konečně mnoha krocích a snadno ověříme, že je tablo sporné, jinak ale skončit nemusí.

Je-li T kompletní, víme, že platí právě jedna z následujících možností: buď $T \vdash \varphi$ nebo $T \vdash \neg\varphi$. Budeme tedy paralelně konstruovat tablo pro Fφ a tablo pro Tφ (důkaz a zamítnutí φ z T): jedna z konstrukcí po konečně mnoha krocích skončí. □

10.1.1 Rekurzivně spočetná kompletace

Požadavek kompletnosti je příliš silný, ukážeme, že stačí pokud jsme schopni efektivně popsat všechny kompletní jednoduché extenze.[5]

Definice 10.1.5 (Rekurzivně spočetná kompletace)**.** Řekneme, že teorie T má *rekurzivně spočetnou kompletaci*, pokud (nějaká) množina až na ekvivalenci všech jednoduchých kompletních extenzí teorie T je *rekurzivně spočetná*, tj. existuje algoritmus, který pro danou vstupní dvojici přirozených čísel (i, j) vypíše na výstup i-tý axiom j-té extenze (v nějakém pevně daném uspořádání[6]), nebo odpoví, že takový axiom už neexistuje.[7]

Tvrzení 10.1.6. *Pokud je teorie T rekurzivně axiomatizovaná a má rekurzivně spočetnou kompletaci, potom je T rozhodnutelná.*

[4] Zde nám stačí enumerátor axiomů T, nebo postupně generujeme všechny sentence (např. v lexikografickém pořadí) a pro každou testujeme, zda je axiomem.

[5] Tj. 'všechny modely až na elementární ekvivalenci'.

[6] Zde potřebujeme, aby byl jazyk spočetný.

[7] Jeli extenzí méně než j, nebo má-li j-tá extenze méně než i axiomů.

Důkaz. Pro danou sentenci φ buď $T \vdash \varphi$, nebo existuje protipříklad $\mathcal{A} \models \varphi$, tedy kompletní jednoduchá extenze T_i teorie T taková, že $T_i \not\vdash \varphi$. Z kompletnosti ale plyne, že $T_i \vdash \neg\varphi$. Náš algoritmus bude paralelně konstruovat tablo důkaz φ z T a (postupně) tablo důkazy $\neg\varphi$ ze všech kompletních jednoduchých extenzí $T_1, T_2, \ldots$ teorie T.[8] Víme, že alespoň jedno z paralelně konstruovaných tabel je sporné, a můžeme předpokládat, že konečné (neprodlužujeme-li sporné větve tabla), tedy algoritmus ho po konečně mnoha krocích zkonstruuje. □

Cvičení 10.1. Ukažte, že následující teorie mají rekurzivně spočetnou kompletaci:

- Teorie čisté rovnosti (prázdná teorie v jazyce $L = \langle\rangle$ s rovností),
- Teorie unárního predikátu (prázdná teorie v jazyce $L = \langle U \rangle$ s rovností, kde U je unární relační symbol),
- Teorie hustých lineárních uspořádání DeLO* (kompletní jednoduché extenze jsou popsané v Důsledku 9.3.5),

Jde o rekurzivně axiomatizované teorie (neboť jsou konečné), jsou tedy rozhodnutelné.

Příklad 10.1.7. Na závěr uveďme bez důkazu několik dalších příkladů rozhodnutelných teorií:

- Teorie Booleových algeber (Alfred Tarski 1940),
- Teorie algebraicky uzavřených těles (Tarski 1949),
- Teorie komutativních grup (Wanda Szmielew 1955).

Tyto teorie jsou také nekompletní, ale rekurzivně axiomatizované a mají rekurzivně spočetnou kompletaci.

10.1.2 Rekurzivní axiomatizovatelnost

V předchozí kapitole, konkrétně v Sekci 9.4, jsme se zabývali otázkou, kdy lze popsat nějakou třídu struktur [resp. teorii] pomocí axiomů [určitého tvaru]. Nyní se zaměřme na otázku, kdy to lze udělat *algoritmicky*.

Definice 10.1.8 (Rekurzivní axiomatizovatelnost)**.** Třída L-struktur $K \subseteq \mathrm{M}_L$ je *rekurzivně axiomatizovatelná*, pokud existuje rekurzivně axiomatizovaná L-teorie T taková, že $K = M_L(T)$. Teorie T' je *rekurzivně axiomatizovatelná*, pokud je rekurzivně axiomatizovatelná třída jejích modelů, neboli pokud je T' ekvivalentní nějaké rekurzivně axiomatizované teorii.

Poznámka 10.1.9. Podobně bychom mohli definovat *rekurzivně spočetnou axiomatizovatelnost*.

Ukažme si následující jednoduché tvrzení:

Tvrzení 10.1.10. *Je-li $\mathcal{A}$ konečná struktura v konečném jazyce s rovností, potom je teorie* $\mathrm{Th}(\mathcal{A})$ *rekurzivně axiomatizovatelná.*

Poznámka 10.1.11. Z toho plyne i že $\mathrm{Th}(\mathcal{A})$ je rozhodnutelná, což ale není překvapivé: platnost sentence φ v konečné struktuře $\mathcal{A}$ můžeme snadno ověřit.

[8] Nevadí, že je jich nekonečně mnoho, můžeme využít tzv. *dovetailing*: Provedeme 1. krok konstrukce 1. tabla, potom 2. krok 1. tabla a 1. krok 2. tabla, 3. krok 1. tabla, 2. krok 2. tabla, 1. krok 3. tabla, atd.

Důkaz. Očíslujme prvky domény jako $A = \{a_1, \ldots, a_n\}$. Teorii $\mathrm{Th}(\mathcal{A})$ lze axiomatizovat jedinou sentencí, která je tvaru "existuje právě n prvků $a_1, \ldots, a_n$ splňujících právě ty *základní vztahy* o funkčních hodnotách a relacích, které platí ve struktuře $\mathcal{A}$".[9] □

Uveďme několik standardních příkladů struktur, které lze 'algoritmicky popsat':

Příklad 10.1.12. Pro následující struktury je $\mathrm{Th}(\mathcal{A})$ rekurzivně axiomatizovatelná, a tedy i rozhodnutelná:

- $\langle \mathbb{Z}, \leq \rangle$, jde o tzv.teorii *diskrétních lineárních uspořádání*,
- $\langle \mathbb{Q}, \leq \rangle$, jde o teorii DeLO,
- $\langle \mathbb{N}, S, 0 \rangle$, teorie *následníka s nulou*,
- $\langle \mathbb{N}, S, +, 0 \rangle$, *Presburgerova aritmetika*,
- $\langle \mathbb{R}, +, -, \cdot, 0, 1 \rangle$, teorie *reálně uzavřených těles*,[10]
- $\langle \mathbb{C}, +, -, \cdot, 0, 1 \rangle$, teorie *algebraicky uzavřených těles charakteristiky 0.*

Důsledek 10.1.13. *Pro struktury uvedené v Příkladu 10.1.12 platí, že* $\mathrm{Th}(\mathcal{A})$ *je rozhodnutelná.*

Poznámka 10.1.14. Jak ale vyplývá z První Gödelovy věty o neúplnosti (viz níže), *standardní model aritmetiky*, tj. struktura $\underline{\mathbb{N}} = \langle \mathbb{N}, S, +, \cdot, 0, \leq \rangle$, *nemá* rekurzivně axiomatizovatelnou teorii.

10.2 Aritmetika

Vlastnosti přirozených čísel hrají důležitou roli nejen v matematice, ale například také v kryptografii. Připomeňme, že jazyk aritmetiky je jazyk $L = \langle S, +, \cdot, 0, \leq \rangle$ s rovností. Jak jsme zmínili v Poznámce 10.1.14, tzv. *standardní model aritmetiky* $\underline{\mathbb{N}} = \langle \mathbb{N}, S, +, \cdot, 0, \leq \rangle$ nemá rekurzivně axiomatizovatelnou teorii. Proto používáme rekurzivně axiomatizované teorie, které se snaží vlastnosti $\underline{\mathbb{N}}$ popsat částečně; těmto teoriím říkáme *aritmetiky.*

10.2.1 Robinsonova a Peanova aritmetika

Uvedeme jen dva nejdůležitější příklady aritmetik: *Robinsonovu* a *Peanovu.*

Definice 10.2.1 (Robinsonova aritmetika)**.** *Robinsonova aritmetika* je teorie Q v jazyce aritmetiky sestávající z následujících (konečně mnoha) axiomů:

$$\begin{array}{ll} \neg S(x) = 0 & x \cdot 0 = 0 \\ S(x) = S(y) \to x = y & x \cdot S(y) = x \cdot y + x \\ x + 0 = x & \neg x = 0 \to (\exists y)(x = S(y)) \\ x + S(y) = S(x + y) & x \leq y \leftrightarrow (\exists z)(z + x = y) \end{array}$$

[9] Například, pokud $f^{\mathcal{A}}(a_4, a_2) = a_{17}$, přidáme do konjunkce atomickou formuli $f(x_{a_4}, x_{a_2}) = x_{a_{17}}$ (kde x_{a_i} jsou proměnné odpovídající jednotlivým prvkům). A pokud $(a_3, a_3, a_1) \notin R^{\mathcal{A}}$, přidáme $\neg R(x_{a_3}, x_{a_3}, x_{a_1})$.

[10] Tento významný výsledek A. Tarského (1949) také znamená, že lze algoritmicky rozhodovat, které vlastnosti platí v Euklidovské geometrii.

Robinsonova aritmetika je velmi slabá, nelze v ní dokázat například komutativitu ani asociativitu sčítání či násobení, nebo tranzitivitu uspořádání.

Na druhou stranu v ní lze dokázat všechna *existenční tvrzení o numerálech*, která jsou pravdivá v $\underline{\mathbb{N}}$. Tím myslíme formule, které v prenexním tvaru mají pouze existenční kvantifikátory, a do kterých jsme za volné proměnné substituovali *numerály* $\underline{n} = S(\dots S(0)\dots)$.

Příklad 10.2.2. Například, pro formuli $\varphi(x, y)$ tvaru $(\exists z)(x + z = y)$ je $Q \vdash \varphi(\underline{1}, \underline{2})$, kde $\underline{1} = S(0)$ a $\underline{2} = S(S(0))$.

Platí tedy následující tvrzení, které ponecháme bez důkazu.

Tvrzení 10.2.3. *Je-li $\varphi(x_1, \dots, x_n)$ existenční formule a $a_1, \dots, a_n \in \mathbb{N}$, potom platí:*

$$Q \vdash \varphi(x_1/\underline{a_1}, \dots, x_n/\underline{a_n}) \text{ právě když } \underline{\mathbb{N}} \models \varphi[e(x_1/a_1, \dots, x_n/a_n)]$$

Užitečným rozšířením Robinsonovy aritmetiky je tzv. Peanova aritmetika, ve které lze *dokazovat indukcí*:

Definice 10.2.4 (Peanova aritmetika)**.** *Peanova aritmetika PA* je extenze Robinsonovy aritmetiky Q o *schéma indukce*, tj. pro každou L-formuli $\varphi(x, \overline{y})$ přidáme následující axiom:

$$(\varphi(0, \overline{y}) \wedge (\forall x)(\varphi(x, \overline{y}) \to \varphi(S(x), \overline{y}))) \to (\forall x)\varphi(x, \overline{y})$$

Peanova aritmetika je mnohem lepší aproximací teorie $\mathrm{Th}(\underline{\mathbb{N}})$, lze v ní dokázat všechny 'základní' vlastnosti platné v $\underline{\mathbb{N}}$ (například komutativitu a asociativitu sčítání). Stále ale existují sentence v jazyce aritmetiky, které platí v $\underline{\mathbb{N}}$, ale v Peanově aritmetice jsou nezávislé.[11]

Poznámka 10.2.5. Pokud bychom se přesunuli do logiky *2. řádu*, potom bychom už mohli strukturu $\underline{\mathbb{N}}$ axiomatizovat (až na izomorfismus), a to extenzí Peanovy aritmetiky o následující formuli 2. řádu, tzv. *axiom indukce*:

$$(\forall X)((X(0) \wedge (\forall x)(X(x) \to X(S(x)))) \to (\forall x)X(x))$$

Připomeňme, že X reprezentuje (libovolnou) unární relaci, neboli podmnožinu univerza. Použitím axiomu indukce na množinu následníků 0 získáme, že každý prvek (daného modelu) je následníkem nuly. Tak můžeme sestrojit izomorfismus s $\underline{\mathbb{N}}$.

10.3 Nerozhodnutelnost predikátové logiky

V této sekci si ukážeme, že nelze (algoritmicky) rozhodovat logickou platnost formulí prvního řádu. (Jinými slovy, nerozhodnutelnost prázdné teorie nad jazykem daným na vstupu.)

Věta 10.3.1 (O nerozhodnutelnosti predikátové logiky)**.** *Neexistuje algoritmus, který by pro danou vstupní formuli φ rozhodl, zda je logicky platná.*[12]

Protože zatím neznáme potřebný formalismus týkající se algoritmů, např. pojem Turingova stroje, zvolíme jako výchozí bod jiný *nerozhodnutelný problém*. Nejznámějším je tzv. *Halting problem*, tj. otázka, zda se daný program zastaví na daném vstupu.[13] My si ale usnadníme práci tím, že zvolíme jiný nerozhodnutelný problém, tzv. *Hilbertův desátý problém*.[14]

[11] Jak si ukážeme v Gödelově První větě o neúplnosti.

[12] Tj. zda je formule φ tautologie, neboli zda $\models \varphi$. Zde mluvíme o formulích 1. řádu, v libovolném jazyce.

[13] Jeho nerozhodnutelnost si dokážete v předmětech NTIN071 Automaty a gramatiky a poté znovu v NTIN090 Základy složitosti a vyčíslitelnosti.

[14] Hilbert jej vyslovil v roce 1900, a publikoval v roce 1902 spolu s 22 dalšími problémy, které významně ovlivnily matematiku 20., i 21. století. Některé zůstávají nevyřešeny, např. Riemannova hypotéza, viz Wikipedia.

10.3.1 Hilbertův desátý problém

Mějme polynom $p(x_1, \ldots, x_n)$ s celočíselnými koeficienty. Hilbertův desátý problém se ptá po algoritmu, který rozhodne, zda má takový vstupní polynom celočíselný kořen, neboli zda má *Diofantická rovnice* $p(x_1, \ldots, x_n) = 0$ (celočíselné) řešení:

> "Nalezněte algoritmus, který po konečně mnoha krocích určí, zda daná Diofantická rovnice s libovolným počtem proměnných a celočíselnými koeficienty má celočíselné řešení."

Kdyby se Hilbert dožil vyřešení svého desátého problému v roce 1970, byl by překvapen, že žádný takový algoritmus neexistuje.

Věta 10.3.2 (Matiyasevich, Davis, Putnam, Robinson)**.** *Problém existence celočíselného řešení dané Diofantické rovnice s celočíselnými koeficienty je (algoritmicky) nerozhodnutelný.*

Důkaz zde pro nedostatek místa neuvedeme. K důkazu nerozhodnutelnosti ve skutečnosti použijeme následující důsledek, který mluví o polynomech s přirozenými koeficienty, a o řešení v přirozených číslech.

Důsledek 10.3.3. *Neexistuje algoritmus, který by pro danou dvojici polynomů $p(x_1, \ldots, x_n)$, $q(x_1, \ldots, x_n)$ s* přirozenými *koeficienty rozhodl, zda mají přirozené řešení, tj. zda platí:*

$$\underline{\mathbb{N}} \models (\exists x_1) \ldots (\exists x_n)\ p(x_1, \ldots, x_n) = q(x_1, \ldots, x_n)$$

Důkaz důsledku. Důkaz je snadný, využívá faktu, že každé celé číslo lze vyjádřit jako rozdíl dvojice přirozených čísel, a naopak, každé přirozené číslo lze vyjádřit jako součet čtyř čtverců (celých čísel).[15] Každou Diofantickou rovnici lze tedy transformovat na rovnici z důsledku, a naopak. □

10.3.2 Důkaz nerozhodnutelnosti

Připomeňme, že Robinsonova aritmetika Q má jen konečně mnoho axiomů, $\underline{\mathbb{N}}$ je jejím modelem, a lze v ní dokázat všechna *existenční tvrzení o numerálech* platná v $\underline{\mathbb{N}}$. Nyní jsme připraveni dokázat Větu o nerozhodnutelnosti predikátové logiky.

Důkaz věty o nerozhodnutelnosti predikátové logiky. Uvažme formuli φ tvaru

$$(\exists x_1) \ldots (\exists x_n)\ p(x_1, \ldots, x_n) = q(x_1, \ldots, x_n)$$

kde p a q jsou polynomy s přirozenými koeficienty. Dle Tvrzení 10.2.3 platí:

$$\underline{\mathbb{N}} \models \varphi \text{ právě když } Q \vdash \varphi$$

Označme jako ψ_Q konjunkci (generálních uzávěrů) všech axiomů Q. Zřejmě $Q \vdash \varphi$, právě když $\psi_Q \vdash \varphi$, což platí právě když $\vdash \psi_Q \to \varphi$. Dle Věty o úplnosti je to ale ekvivalentní $\models \psi_Q \to \varphi$. Dostáváme tedy následující ekvivalenci:

$$\underline{\mathbb{N}} \models \varphi \text{ právě když } \models \psi_Q \to \varphi$$

To znamená, že pokud existoval algoritmus rozhodující logickou platnost, mohli bychom rozhodovat i existenci přirozeného řešení rovnice $p(x_1, \ldots, x_n) = q(x_1, \ldots, x_n)$, neboli Hilbertův desátý problém by byl rozhodnutelný.[16] Což by byl spor. □

[15] Tzv. Lagrangeova věta o čtyřech čtvercích.

[16] Ukazujeme, že existuje *redukce* 'těžkého' problému (Hilbertova desátého) na náš problém, tedy i náš problém je 'těžký'.

10.4 Gödelovy věty

Na závěr přednášky představíme slavné Gödelovy věty o neúplnosti, jejichž pochopení by mělo být samozřejmou součástí vzdělání každého informatika. Pokusíme se vysvětlit i princip důkazů, ale vynecháme veškeré technické detaily.

10.4.1 První věta o neúplnosti

Nejprve vyslovíme Gödelovu *První větu o neúplnosti*, a vysvětlíme smysl jejích předpokladů.

Věta 10.4.1 (První věta o neúplnosti)**.** *Pro každou bezespornou rekurzivně axiomatizovanou extenzi T Robinsonovy aritmetiky existuje sentence, která je pravdivá v $\underline{\mathbb{N}}$, ale není dokazatelná v T.*

Takové sentenci se říká *Gödelova sentence*. Velmi neformálně řečeno, Gödelova První věta o neúplnosti říká, že vlastnosti aritmetiky přirozených čísel nelze 'rozumně', efektivně popsat (v logice 1. řádu), každý takový popis je nutně 'neúplný'. Je důležité si uvědomit, že mluvíme o *pravdivosti* ve standardním modelu aritmetiky, tj. ve struktuře $\underline{\mathbb{N}}$, zatímco *dokazatelnost* je v teorii T. (Z Věty o úplnosti samozřejmě plyne, že každá sentence *pravdivá v T* je v T i dokazatelná.)

Bezespornost je nutným předpokladem, neboť ve sporné teorii je dokazatelná každá sentence. Připomeňme, že rekurzivní axiomatizovanost můžeme chápat jako 'efektivní zadání' axiomů (pomocí algoritmu), bez této vlastnosti by taková teorie nebyla užitečná. Požadavek aby teorie byla extenzí Robinsonovy aritmetiky chápejte jako předpoklad, že má alespoň 'základní aritmetickou sílu', že v ní lze určitým způsobem 'mluvit' o přirozených číslech. Existují různé varianty tohoto předpokladu, s jinými teoriemi než je Robinsonova aritmetika, a není například nutné, aby šlo přímo o extenzi, stačí, když je v teorii Robinsonova aritmetika v jistém smyslu 'definovatelná'. Ale teorie, ve které 'nelze zakódovat přirozená čísla' (a zde je důležité, že můžeme mluvit nejen o sčítání, ale i o násobení), je 'příliš slabá'.

Je dobré si uvědomit, že speciálně platí i následující tvrzení 'o nekompletnosti':

Důsledek 10.4.2. *Splňuje-li teorie T předpoklady První věty o neúplnosti a je-li navíc $\underline{\mathbb{N}}$ modelem teorie T, potom T není kompletní.*

Důkaz. Předpokládejme pro spor, že T je kompletní. Vezměme sentenci φ, která je pravdivá v $\underline{\mathbb{N}}$ ale není dokazatelná v T. Díky kompletnosti víme, že $T \vdash \neg\varphi$, potom ale Věta o korektnosti říká, že $T \models \neg\varphi$, tedy φ je lživá v $\underline{\mathbb{N}}$, což je spor. □

Zajímavé je nejen samotné tvrzení První věty o neúplnosti, ale také její důkaz: Gödel v něm přišel se zcela novou, na svou dobu převratnou důkazovou technikou. Sentence sestrojená v důkazu formalizuje tvrzení *"Nejsem dokazatelná v T"*, důkaz je založen na následujících dvou principech, které níže poněkud neformálně popíšeme:

- *aritmetizace syntaxe*, tedy zakódování sentencí a jejich dokazatelnosti do přirozených čísel,
- *self-reference*, tedy schopnost sentence 'mluvit sama o sobě' (o svém kódu).

Aritmetizace dokazatelnosti

Konečné syntaktické objekty, jako jsou termy, formule, konečná tabla, a tedy i tablo důkazy, lze 'rozumně' zakódovat do přirozených čísel.[17] Konkrétní způsob jak to lze provést, tzv. *Gödelovo číslování*, jako technický detail přeskočíme. Stačí nám, že jsme schopni objekty 'algoritmicky' kódovat a dekódovat (případně 'simulovat manipulaci s objekty' na jejich kódech).

Označme kód formule φ jako $\ulcorner\varphi\urcorner$, podobně pro jiné syntaktické objekty. Numerál odpovídající kódu φ, tedy $\ulcorner\varphi\urcorner$-tý numerál, budeme značit $\underline{\varphi}$. Pro danou teorii T definujme následující binární relaci na množině všech přirozených čísel:

$$(n, m) \in \mathrm{Proof}_T \quad \text{právě když} \quad n = \ulcorner\varphi\urcorner \text{ a } m = \ulcorner\tau\urcorner, \text{ kde } \tau \text{ je tablo důkaz sentence } \varphi \text{ z } T$$

Máme-li efektivní přístup k axiomům, umíme také efektivně zkontrolovat zda τ je opravdu důkazem φ (kde τ a φ získáme dekódováním m a n), tedy platí:

Pozorování 10.4.3. *Je-li T rekurzivně axiomatizovaná, je relace* $\mathrm{Proof}_T \subseteq \mathbb{N}^2$ rekurzivní.

Klíčovou, ale velmi technickou částí důkazu První věty je následující tvrzení, které ponecháme bez důkazu.

Tvrzení 10.4.4. *Je-li T navíc extenzí Robinsonovy aritmetiky Q, potom existuje formule $Prf_T(x, y)$ v jazyce aritmetiky, která* reprezentuje *relaci* Proof_T*, tj. pro každá $n, m \in \mathbb{N}$ platí:*

- *Je-li $(n, m) \in \mathrm{Proof}_T$, potom $Q \vdash Prf_T(\underline{n}, \underline{m})$,*
- *jinak $Q \vdash \neg Prf_T(\underline{n}, \underline{m})$.*

Formule $Prf_T(x, y)$ tedy vyjadřuje *"y je důkaz x v T"*.[18] Potom můžeme vyjádřit, že *"x je dokazatelná v T"*, a to formulí $(\exists y) Prf_T(x, y)$. Všimněte si, že platí následující pozorování, neboť svědek poskytuje kód nějakého tablo důkazu, a $\underline{\mathbb{N}}$ splňuje axiomy Q:

Pozorování 10.4.5. *$T \vdash \varphi$ právě když $\underline{\mathbb{N}} \models (\exists y) Prf_T(\underline{\varphi}, y)$.*

Budeme potřebovat i následující důsledek, který vyslovíme také bez důkazu:

Důsledek 10.4.6 (O predikátu dokazatelnosti)**.** *Je-li $T \vdash \varphi$, potom $T \vdash (\exists y) Prf_T(\underline{\varphi}, y)$.*

Umíme tedy vyjádřit, že daná sentence je, nebo není, dokazatelná. Jak ale může sentence říci 'sama o sobě', že není dokazatelná? K tomu použijeme *princip self-reference*.

Self-reference

Abychom ilustrovali princip self-reference, pro názornost si místo logické sentence představme větu v češtině, a místo vlastnosti "být dokazatelný" tvrzení o počtu písmen. Podívejme se na následující větu:

`Tato věta má 22 znaků.`

[17] Představte si jakýkoliv rozumný způsob, jak daný objekt zapsat do souboru. Soubor v binárním kódu je posloupnost 0 a 1. Připíšeme na začátek jedničku, abychom nezačínali nulou, a máme binární zápis přirozeného čísla.

[18] Přesněji, tablo jehož kódem je y je důkazem sentence jejíž kódem je x.

V přirozeném jazyce snadno vyjádříme self-referenci zájmenem "Tato", z kontextu víme, že myslíme větu samou. Ve formálních systémech ale typicky nemáme self-referenci přímo k dispozici. *Přímou referenci* obvykle máme k dispozici, stačí umět 'mluvit' o posloupnostech symbolů, jako v následujícím příkladě:

```
Následující věta má 29 znaků. "Následující věta má 29 znaků."
```

Zde se ale není žádná self-reference. Pomůžeme si trikem, kterému budeme říkat 'zdvojení':

```
Následující věta zapsaná jednou a ještě jednou v uvozovkách má 149
znaků. "Následující věta zapsaná jednou a ještě jednou v uvozovkách
má 149 znaků."
```

Pomocí přímé reference a zdvojení tedy můžeme získat self-referenci.

Poznámka 10.4.7. Stejný princip lze použít k sestrojení programu v C, jehož výstupem je jeho vlastní kód (34 je ASCII kód uvozovek):

```
main(){char *c="main(){char *c=%c%s%c; printf(c,34,c,34);}"; printf(c,34,c,34);}
```

10.4.2 Důkaz a důsledky

V této podsekci dokážeme První Gödelovu větu o neúplnosti a řekneme si i něco o jejích důsledcích. Budeme potřebovat následující větu, která popisuje, jak technicky využijeme princip self-reference. Lze na ní nahlížet jako na formu 'diagonalizačního argumentu',[19] proto se tomuto tvrzení také někdy říká *diagonální lemma*.

Věta 10.4.8 (Věta o pevném bodě)**.** *Je-li T extenzí Robinsonovy aritmetiky, potom pro každou formuli $\varphi(x)$ (v jazyce teorie T) existuje sentence ψ taková, že platí:*

$$T \vdash \psi \leftrightarrow \varphi(\underline{\psi})$$

Sentence ψ je tedy *self-referenční*, říká o sobě: "splňuji vlastnost φ".[20] Vysvětlíme si jen myšlenku důkazu. Všimněte si, jak se v důkazu použije přímá reference a zdvojení.

Důkaz. Uvažme *zdvojující funkci*, funkci $d\colon \mathbb{N} \to \mathbb{N}$ takovou, že pro každou formuli $\chi(x)$ platí:

$$d(\ulcorner \chi(x) \urcorner) = \ulcorner \chi(\underline{\chi(x)}) \urcorner$$

Funkce d tedy dostane na vstupu přirozené číslo n, které dekóduje jako formuli v jedné proměnné, dosadí do této formule numerál $\underline{n}$,[21] a výslednou sentenci znovu zakóduje.

S využitím předpokladu, že T je extenzí Q, lze dokázat, že tato funkce je v T *reprezentovatelná*. Pro jednoduchost předpokládejme, že je reprezentovatelná termem,[22] a označme ho také d. To znamená, že pro každou formuli $\chi(x)$ platí:

$$T \vdash d(\underline{\chi(x)}) = \underline{\chi(\underline{\chi(x)})}$$

[19] Diagonalizací se myslí argument připomínající *Cantorův diagonální argument*, známý z důkazu nespočetnosti $\mathbb{R}$. Podobný argument, používající self-referenci, potkáme třeba v *Holičově paradoxu*, nebo v důkazu nerozhodnutelnosti *Halting problému*.

[20] Přesněji, říká to o numerálu odpovídajícímu jejímu kódu.

[21] Zde *numerál* odpovídá 'uvozovkám' z předchozího neformálního popisu self-reference, a $d(\ulcorner \chi \urcorner)$ znamená "χ napsaná jednou a ještě jednou v uvozovkách."

[22] Byť ve skutečnosti je reprezentovaná (složitou) formulí.

Tedy Robinsonova aritmetika, a tím pádem i naše teorie T, dokazuje *o numerálech*, že d opravdu 'zdvojuje'.

Hledaná self-referenční sentence ψ je sentence:[23]

$$\varphi(d(\underline{\varphi(d(x))}))$$

Chceme dokázat, že platí $T \vdash \psi \leftrightarrow \varphi(\underline{\psi})$, neboli $T \vdash \varphi(d(\underline{\varphi(d(x))})) \leftrightarrow \varphi(\underline{\varphi(d(\underline{\varphi(d(x))}))})$. K tomu stačí ověřit, že:

$$T \vdash d(\underline{\varphi(d(x))}) = \underline{\varphi(d(\underline{\varphi(d(x))}))}$$

To ale víme z reprezentovatelnosti d, kde za formuli $\chi(x)$ dosadíme $\varphi(d(x))$. □

Než přistoupíme k samotnému důkazu Gödelovy věty, ukážeme si jako rozcvičku jeden důsledek Věty o pevném bodě: *Definicí pravdy* v aritmetické teorii T myslíme formuli $\tau(x)$ takovou, že pro každou sentenci ψ platí:

$$T \vdash \psi \leftrightarrow \tau(\underline{\psi})$$

Pokud by definice pravdy existovala, znamenalo by to, že místo dokazování sentence stačí spočíst její kód, substituovat příslušný numerál do τ, a vyhodnotit.

Věta 10.4.9 (Nedefinovatelnost pravdy)**.** *V žádném bezesporném rozšíření Robinsonovy aritmetiky neexistuje definice pravdy.*

Důkaz využívá *Paradox lháře*, vyjádříme větu "Nejsem pravdivá v T".

Důkaz. Předpokládejme pro spor, že existuje definice pravdy $\tau(x)$. Použijeme Větu o pevném bodě, kde za formuli $\varphi(x)$ vezmeme $\neg\tau(x)$. Dostáváme existenci sentence ψ takové, že:

$$T \vdash \psi \leftrightarrow \neg\tau(\underline{\psi})$$

Protože $\tau(x)$ je definice pravdy, platí ale i $T \vdash \psi \leftrightarrow \tau(\underline{\psi})$, tedy i $T \vdash \tau(\underline{\psi}) \leftrightarrow \neg\tau(\underline{\psi})$. To by ale znamenalo, že T je sporná. □

Důkaz Gödelovy věty používá tentýž trik, ale pro větu "Nejsem dokazatelná v T".

Důkaz První věty o neúplnosti. Mějme bezespornou rekurzivně axiomatizovanou extenzi T Robinsonovy aritmetiky. Chceme najít Gödelovu sentenci ψ_T, která je pravdivá v $\underline{\mathbb{N}}$, ale není dokazatelná v T.

Takovou sentenci získáme z Věty o pevném bodě jako sentenci vyjadřující "Nejsem dokazatelná v T". Nechť $\varphi(x)$ je formule $\neg(\exists y)Prf_T(x, y)$ ("x není dokazatelná v T"). Podle Věty o pevném bodě existuje sentence ψ_T splňující:

$$T \vdash \psi_T \leftrightarrow \neg(\exists y)Prf_T(\underline{\psi_T}, y)$$

Sentence ψ_T je tedy v T ekvivalentní sentenci, která vyjadřuje, že ψ_T není dokazatelná v T. Lze ukázat, že stejná ekvivalence platí i v $\underline{\mathbb{N}}$ (neboť tak jsme Prf_T a ψ_T zkonstruovali):

$$\underline{\mathbb{N}} \models \psi_T \quad \text{právě když} \quad \underline{\mathbb{N}} \models \neg(\exists y)Prf_T(\underline{\psi_T}, y)$$

[23] Následující věta zapsaná jednou a ještě jednou v uvozovkách má vlastnost φ. "Následující věta zapsaná jednou a ještě jednou v uvozovkách má vlastnost φ."

Z Pozorování 10.4.5 získáváme, že

$$\underline{\mathbb{N}} \models \psi_T \text{ právě když } T \nvdash \psi_T$$

neboli ψ_T je pravdivá v $\underline{\mathbb{N}}$, právě když není dokazatelná v T. Stačí tedy ukázat, že ψ_T není dokazatelná v T. Předpokládejme pro spor, že $T \vdash \psi_T$. Ze self-reference víme, že platí $T \vdash \neg(\exists y)Prf_T(\underline{\psi_T}, y)$. Z Důsledku 10.4.6 o predikátu dokazatelnosti ale dostáváme $T \vdash (\exists y)Prf_T(\underline{\psi_T}, y)$, což by znamenalo, že T je sporná. □

Na závěr si ukážeme dva důsledky a jedno zesílení. Následující okamžitý důsledek už jsme zmínili dříve:

Důsledek 10.4.10. *Je-li T rekurzivně axiomatizovaná extenze Robinsonovy aritmetiky a je-li navíc $\underline{\mathbb{N}}$ modelem teorie T, potom T není kompletní.*

Důkaz. Protože má T model, není sporná. Splňuje tedy předpoklady První věty o neúplnosti, tedy v ní není dokazatelná Gödelova sentence ψ_T. Pokud by byla kompletní, musela by dokazovat $\neg\psi_T$. To by ale znamenalo, že platí i $\underline{\mathbb{N}} \models \neg\psi_T$, přičemž víme, že ψ_T je v $\underline{\mathbb{N}}$ pravdivá. □

Z toho plyne, že nelze rekurzivně axiomatizovat standardní model přirozených čísel:

Důsledek 10.4.11. *Teorie* $\mathrm{Th}(\underline{\mathbb{N}})$ *není rekurzivně axiomatizovatelná.*

Důkaz. Teorie $\mathrm{Th}(\underline{\mathbb{N}})$ je extenzí Robinsonovy aritmetiky a platí v modelu $\underline{\mathbb{N}}$. Pokud by byla rekurzivně axiomatizovatelná, její (libovolná) rekurzivní axiomatizace by podle předchozího důsledku nemohla být kompletní. Ale $\mathrm{Th}(\underline{\mathbb{N}})$ kompletní je. □

Jedním ze zesílení Gödelovy První věty je následující tvrzení, které uvedeme bez důkazu. Ukazuje, že předpoklad $\underline{\mathbb{N}} \models T$ v prvním důsledku výše je ve skutečnosti nadbytečný.

Věta 10.4.12 (Rosserův trik, 1936)**.** *V každé bezesporné rekurzivně axiomatizované extenzi Robinsonovy aritmetiky existuje nezávislá sentence. Tedy taková teorie není kompletní.*

10.4.3 Druhá věta o neúplnosti

Druhá Gödelova věta o neúplnosti říká, neformálně řečeno, že efektivně daná, dostatečně bohatá teorie nemůže sama dokázat svou bezespornost. Bezespornost ("konzistenci") vyjádříme následující sentencí, kterou označíme jako Con_T:

$$\neg(\exists y)Prf_T(\underline{0 = S(0)}, y)$$

Všimněte si, že platí $\underline{\mathbb{N}} \models Con_T$, právě když $T \nvdash 0 = S(0)$. Neboli sentence Con_T opravdu vyjadřuje, že *"Teorie T je bezesporná"*.

Věta 10.4.13 (Druhá věta o neúplnosti)**.** *Pro každou bezespornou rekurzivně axiomatizovanou extenzi T Peanovy aritmetiky platí, že Con_T není dokazatelná v T.*

Všimněte si, že sentence Con_T je přitom pravdivá v $\underline{\mathbb{N}}$ (neboť T je opravdu bezesporná). Zmiňme také, že není třeba plná síla Peanovy aritmetiky, stačí slabší předpoklad. Nyní si ukážeme hlavní myšlenku důkazu Druhé věty:

Důkaz Druhé věty o neúplnosti. Vezměme Gödelovu sentenci ψ_T vyjadřující "nejsem dokazatelná v T". V důkazu První věty o neúplnosti (konkrétně v první části) jsme ukázali, že:

"Pokud je T bezesporná, potom ψ_T není dokazatelná v T."

Z toho jednak plyne, že $T \nvdash \psi_T$, neboť T bezesporná je. Na druhou stranu to lze formulovat jako "platí $Con_T \to \psi_T$" a je-li T extenze Peanovy aritmetiky, lze důkaz tohoto tvrzení zformalizovat v rámci teorie T, tedy ukázat, že:

$$T \vdash Con_T \to \psi_T$$

Kdyby platilo $T \vdash Con_T$, dostali bychom i $T \vdash \psi_T$, což by byl spor. □

Na závěr si ukážeme tři důsledky Druhé věty.

Důsledek 10.4.14. *Existuje model PA, ve kterém platí sentence* $(\exists y) Prf_{PA}(\underline{0 = S(0)}, y)$.

Důkaz. Sentence Con_{PA} není dokazatelná, tedy ani pravdivá v PA. Platí ale v $\underline{\mathbb{N}}$ (neboť PA je bezesporná), což znamená, že je Con_{PA} nezávislá v PA. V nějakém modelu tedy musí platit její negace, která je ekvivalentní $(\exists y) Prf_{PA}(\underline{0 = S(0)}, y)$.

□

Uvědomme si, že musí jít o nestandardní model PA, svědkem musí být *nestandardní* prvek (tj. takový, který není hodnotou žádného numerálu).

Důsledek 10.4.15. *Existuje bezesporná rekurzivně axiomatizovaná extenze T Peanovy aritmetiky, která 'dokazuje svou spornost', tj. taková, že $T \vdash \neg Con_T$.*

Důkaz. Uvažme teorii $T = PA \cup \{\neg Con_{PA}\}$. Tato teorie je bezesporná, neboť $PA \nvdash Con_{PA}$. Také triviálně platí $T \vdash \neg Con_{PA}$ (tj. T 'dokazuje spornost' teorie PA). Protože je $PA \subseteq T$, platí i $T \vdash \neg Con_T$. □

Zde si uvědomme, že $\underline{\mathbb{N}}$ nemůže být modelem teorie T.

Nakonec se podívejme na teorii ZFC, tj. Zermelovu–Fraenkelovu teorii množin s axiomem výběru, na které je založena formalizace matematiky. Tato teorie není formálně vzato extenzí PA, ale není problém v ní Peanovu aritmetiku (v jistém smyslu) 'interpretovat'. To znamená, že ani tato teorie neumí dokázat svou vlastní bezespornost.

Důsledek 10.4.16. *Je-li teorie množin ZFC bezesporná, nemůže být sentence Con_{ZFC} v teorii ZFC dokazatelná.*

Pokud by tedy někdo v rámci teorie ZFC dokázal, že je ZFC bezesporná, znamenalo by to, že je ZFC sporná. Což bude taková pěkná tečka za naší přednáškou.

Literatura

[1] Mordechai Ben-Ari. *Mathematical Logic for Computer Science*. Springer London, June 2012. Google-Books-ID: hxOpugAACAAJ.

[2] Petr Gregor. Výroková a predikátová logika.

[3] Anil Nerode and Richard A. Shore. *Logic for Applications*. Springer Science & Business Media, December 2012. Google-Books-ID: 90HhBwAAQBAJ.

[4] Martin Pilát. Lecture Notes on Propositional and Predicate Logic. original-date: 2017-10-05T20:42:26Z.