ONDŘEJ LENGÁL

ZÁKLADY LOGIKY PRO INFORMATIKY—ČÁST 1

FIT VUT V BRNĚ

1
Výroková logika

1.1 Syntaxe

Před tím, než se pustíme do popisu významu logických formulí, je potřeba určit, jak vůbec logické formule vypadají. Jinými slovy, je potřeba nadefinovat jejich *syntaxi*[1]. S logickými formulemi se čtenář jistě již setkal, takže intuitivně asi již tuší, že zatímco řetězce

- „$(x_1 \wedge x_2) \to x_3$“ nebo
- „$(x \wedge y \wedge \neg x) \vee (\neg x \wedge \neg y \wedge z)$“

značí logické formule, tak řetězce

- „$x \to$“,
- „$\wedge \vee y($“ nebo
- „$x \wedge y \vee z$“[2]

logické formule *ne*značí.

Syntaxe určuje, jak správně zapsat *formule* výrokové logiky. Podobně jako u programovacích jazyků se i u výrokové logiky syntaxe definuje pomocí

1. *abecedy*, tj., množiny symbolů, které se ve formulích mohou vyskytovat a
2. *gramatiky*, tj., pravidel, pomocí nichž můžeme pomocí symbolů z abecedy stavět formule.

Nejdříve si definujme, jak vypadá abeceda výrokové logiky. Uvažujme spočetně nekonečnou množinu[3] *výrokových proměnných* $\mathbb{X} = \{x, y, z, \ldots, x_1, x_2, \ldots\}$. *Abeceda výrokové logiky* je množina $\mathbb{X} \cup \{0, 1, \neg, \wedge, \vee, \to, \leftrightarrow, (,)\}$, kde symbolům $\{\neg, \wedge, \vee, \to, \leftrightarrow\}$ říkáme *logické spojky* a symbolům „0“ a „1“ říkáme *logické konstanty. Formule výrokové logiky* (v této kapitole budeme používat krátce termín *formule*) jsou pak řetězce symbolů, které můžeme nad touto abecedou tvořit pomocí následujících pravidel:

[1] Někdy se také můžete setkat s pojmem *(formální) jazyk*

[2] Není-li definována priorita operátorů $\wedge$ a $\vee$, není jasné, zda má zápis být interpretován jako formule „$(x \wedge y) \vee z$“ nebo jako formule “$x \wedge (y \vee z)$”. Tyto dvě formule mají různou sémantiku.

[3] Množina je *spočetně nekonečná* pokud je nekonečná a pokud její prvky lze jednoznačně očíslovat přirozenými čísly. Příklady spočetně nekonečných množin jsou například množiny přirozených, celých a racionálních čísel $\mathbb{N}$, $\mathbb{Z}$ a $\mathbb{Q}$. Příkladem nespočetně nekonečné množiny je např. množina reálných čísel $\mathbb{R}$. O spočetných a nespočetných množinách se dozvíte více v libovolném textu o teorii množin.

1. Je-li x výroková proměnná, tj. $x \in \mathbb{X}$, pak řetězce „x“, „0“ a „1“ jsou formule.
2. Jsou-li φ a ψ formule, pak jsou formule i řetězce „$(\neg\varphi)$“, „$(\varphi \wedge \psi)$“, „$(\varphi \vee \psi)$“, „$(\varphi \to \psi)$“ a „$(\varphi \leftrightarrow \psi)$“.
3. Formule výrokové logiky jsou právě všechny konečné řetězce získané pomocí předchozích dvou pravidel.

Pro množinu všech formulí výrokové logiky vytvořených pomocí výše uvedených pravidel bude používat označení Φ_{VL}.

Poznámka 1.1

Předchozí definice gramatiky formulí výrokové logiky lze také nahradit zápisem ve stylu Backus-Naurovy formy (BNF) [Bac59, Nau61] známé z definice syntaxe např. programovacích jazyků nebo datových formátů:

$$\varphi ::= x \in \mathbb{X} \mid 0 \mid 1 \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Příklad 1.1

Následující řetězce jsou příklady formulí výrokové logiky:

- „$((x \wedge y) \vee z)$“,
- „$(x_{10} \to (1 \wedge (\neg z)))$“ a
- „$((z \wedge y) \leftrightarrow (z \vee y))$“.

Následující řetězce *ne*jsou formule výrokové logiky:

- „$x \to$ “,
- „$\wedge \vee y($“ a
- „$x \wedge y \vee z$“.

Formule sestavené striktně pomocí výše uvedené gramatiky trpí nepříjemným nadbytkem závorek. Jako příklad uveďme formule

$$(x_1 \wedge (x_2 \wedge (x_3 \wedge x_4))) \text{ a } (((x_1 \wedge x_2) \wedge x_3) \wedge x_4).$$

Jde o dvě různé formule, která ale mají (jak si ukážeme v Sekci 1.2) stejnou sémantiku. Z toho důvodu zavedeme následující konvenci, která určuje, kdy není potřeba závorky psát:

- kolem celé formule:

  $$\text{„}(x \wedge (y \to z))\text{“} \quad \rightsquigarrow \quad \text{„}x \wedge (y \to z)\text{“},$$

- u asociativních spojek[4] $(\wedge, \vee, \leftrightarrow)$:

  $$\text{„}(x \wedge (y \wedge z)) \to w\text{“} \quad \rightsquigarrow \quad \text{„}(x \wedge y \wedge z) \to w\text{“},$$

- kolem negace:

  $$\text{„}(\neg(x \wedge y)) \vee ((\neg x) \leftrightarrow z)\text{“} \quad \rightsquigarrow \quad \text{„}\neg(x \wedge y) \vee (\neg x \leftrightarrow z)\text{“}$$

[4] viz Sekce 1.2

Příklad 1.2
Formule z Příkladu 1.1 můžeme pomocí výše uvedené konvence zapsat následujícím způsobem:

- „$((x \wedge y) \vee z)$“ $\rightsquigarrow$ „$(x \wedge y) \vee z$“,
- „$(x_{10} \to (1 \wedge (\neg z)))$“ $\rightsquigarrow$ „$(x_{10} \to (1 \wedge \neg z))$“ a
- „$((z \wedge y) \leftrightarrow (z \vee y))$“ $\rightsquigarrow$ „$(z \wedge y) \leftrightarrow (z \vee y)$“.

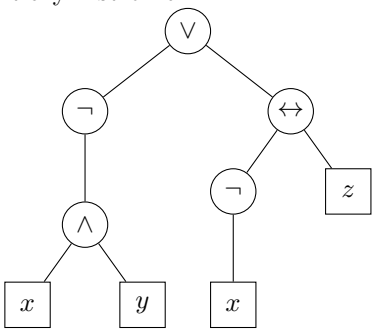
Poznámka 1.2
Někdy se můžete setkat s dalšími pravidly pro vynechávání závorek, například když je dána tzv. *precedence* logických spojek, která určuje, jak „*těsně*“ dané logické spojky vážou své operandy. Precedenci operátorů znáte např. z aritmetiky, kde násobení má vyšší precedenci než sčítání (a lze tedy místo „$x + (y \cdot z)$“ psát „$x + y \cdot z$“) nebo programovacích jazyků, kde je precedence operátorů definována ve standardu jazyka (např. jazyk C++20 má definovaných 17 úrovní precedence). V tomto textu si vystačíme s konvencí definovanou výše.

Způsob, jakým je výše definována množina formulí, se nazývá *induktivní* (někdy též *rekurzivní*) definice. Induktivní definice množiny je zadána pomocí

1. *základních prvků* (v našem případě logických konstant 0, 1 a všech výrokových proměnných $x \in \mathbb{X}$) a
2. *konstrukčních pravidel*, které říkají, jak lze za použití prvků z množiny vytvořit prvky nové (v našem případě jak lze za použití logických spojek vytvořit z existujících formulí novou formuli).

Každý prvek z induktivně definované množiny lze reprezentovat jako tzv. *(abstraktní) syntaktický strom* vytvořený pomocí základních prvků a konstrukčních pravidel.

Příklad 1.3
Formuli „$(\neg(x \wedge y)) \vee ((\neg x) \leftrightarrow z)$“ lze reprezentovat následujícím syntaktickým stromem:



1.2 Sémantika

Sémantika formule určuje její význam, ve výrokové logice tedy to, kdy *platí*. Například formule $(x \wedge y) \vee z$ platí pokud buď (i) proměnné x a y mají hodnotu 1 nebo (ii) proměnná z má hodnotu 1. Pro definici sémantiky je potřeba nejdříve definovat pojem ohodnocení proměnných. *Ohodnocení proměnných* I je zobrazení, které každé proměnné z $\mathbb{X}$[5] přiřadí hodnotu 0 nebo 1[6]. Formálně:

$$I \colon \mathbb{X} \to \{0, 1\} \tag{1.1}$$

Sémantika formule pak určuje, jakou pravdivostní hodnotu formule nabude pro jednotlivá ohodnocení proměnných. Tato hodnota se často definuje induktivně tak, že pro atomickou formuli x odpovídá hodnotě $I(x)$ a pro složenou formuli je definována pomocí tzv. *pravdivostní tabulky*. Pravdivostní tabulka pro dvě formule φ a ψ říká, jaká bude výsledná hodnota formule získané aplikací dané logické spojky na φ a ψ. Pro základní logické spojky vypadá pravdivostní tabulka následovně:

φ	ψ	$\neg\varphi$	$\varphi \vee \psi$	$\varphi \wedge \psi$	$\varphi \to \psi$	$\varphi \leftrightarrow \psi$
0	0	1	0	0	1	1
0	1	1	1	0	1	0
1	0	0	1	0	0	0
1	1	0	1	1	1	1

Příklad 1.4

1. Pravdivostní tabulka pro formuli $x \to (x \wedge \neg y)$ bude vypadat následovně:

x	y	$\neg y$	$x \wedge \neg y$	$x \to (x \wedge \neg y)$
0	0	1	0	1
0	1	0	0	1
1	0	1	1	1
1	1	0	0	0

   Tabulku jsme sestrojili tak, že v levé části jsme vypsali všechny možnosti jak proměnným x a y vyskytujícím se ve formuli přiřadit pravdivostní hodnotu a v pravé části jsme si vypsali všechny podformule dané formule a pro každou spočítali její hodnotu pro všechna možná ohodnocení x a y. Poslední (zvýrazněný) sloupec pak udává, jaká bude hodnota formule pro jednotlivá ohodnocení proměnných (v tomto případě to bude vždy 1 kromě případu, kdy x a y mají obě hodnotu 1).

[5] případně jen *každé proměnné v uvažované formuli*

[6] Místo hodnot 0 a 1 se někdy používají symboly $\bot$ a $\top$, symboly *ff* a *tt*, anglická slova `false` a `true` (například v programovacích jazycích Python, C++, JavaScript, atp.), případně jejich český (poněkud krkolomný) překlad *nepravda* a *pravda*.

2. Pravdivostní tabulka pro formuli φ: $(x \wedge (y \to z)) \leftrightarrow (x \vee \neg z)$ bude vypadat následovně:

x	y	z	$y \to z$	$x \wedge (y \to z)$	$\neg z$	$x \vee \neg z$	φ
0	0	0	1	0	1	1	0
0	0	1	1	0	0	0	1
0	1	0	0	0	1	1	0
0	1	1	1	0	0	0	1
1	0	0	1	1	1	1	1
1	0	1	1	1	0	1	1
1	1	0	0	0	1	1	0
1	1	1	1	1	0	1	1

Poznámka 1.3

Formálně bychom mohli definovat sémantiku formule následujícím způsobem. Nechť $\mathbb{V}$ je množina všech ohodnocení proměnných, tj. $\mathbb{V} = (\mathbb{X} \to \{0,1\})$. Potom sémantika výrokové formule je funkce $[\![\cdot]\!] \colon \mathbb{V} \to \{0,1\}$, např. $[\![\varphi]\!]$, definována induktivně pomocí pravdivostní tabulky.

Je zřejmé, že pravdivostní tabulka pro formuli s n různými proměnnými bude mít 2^n řádků.

Cvičení 1.1

1. Kolik existuje formulí s dvěmi proměnnými?
2. Kolik existuje formulí s proměnnými z množiny $\mathbb{X}$?
3. Kolik existuje formulí s dvěmi proměnnými s *různou sémantikou*? Neboli, kolik existuje různých binárních logických spojek?
4. Kolik existuje formulí s n proměnnými s *různou sémantikou*?

Věta 1.1. *Logické spojky $\wedge, \vee, \leftrightarrow$ jsou asociativní. Logická spojka $\to$ asociativní není.*

Důkaz. Nejprve pomocí pravdivostní tabulky dokážeme asociativitu spojek $\wedge$ a $\vee$.

φ	ψ	χ	$\varphi \wedge \psi$	$(\varphi \wedge \psi) \wedge \chi$	$\psi \wedge \chi$	$\varphi \wedge (\psi \wedge \chi)$	$\varphi \vee \psi$	$(\varphi \vee \psi) \vee \chi$	$\psi \vee \chi$	$\varphi \vee (\psi \vee \chi)$
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	1	1	1	1
0	1	1	0	0	1	0	1	1	1	1
1	0	0	0	0	0	0	1	1	0	1
1	0	1	0	0	0	0	1	1	1	1
1	1	0	1	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Asociativita spojky $\wedge$ plyne z toho, že sloupec pro formuli $(\varphi \wedge \psi) \wedge \chi$ je stejný jako sloupec pro formuli $\varphi \wedge (\psi \wedge \chi)$ (oba sloupce jsou v tabulce zvýrazněny). Podobně pak pro spojku $\vee$ (v levé části tabulky).

Dále stejným způsobem dokážeme asociativitu spojky $\leftrightarrow$ a neasociativitu spojky $\rightarrow$.

φ	ψ	χ	$\varphi \leftrightarrow \psi$	$(\varphi \leftrightarrow \psi) \leftrightarrow \chi$	$\psi \leftrightarrow \chi$	$\varphi \leftrightarrow (\psi \leftrightarrow \chi)$	$\varphi \rightarrow \psi$	$(\varphi \rightarrow \psi) \rightarrow \chi$	$\psi \rightarrow \chi$	$\varphi \rightarrow (\psi \rightarrow \chi)$
0	0	0	1	0	1	0	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1
0	1	0	0	1	0	1	1	0	0	1
0	1	1	0	0	1	0	1	1	1	1
1	0	0	0	1	1	1	0	1	1	1
1	0	1	0	0	0	0	0	1	1	1
1	1	0	1	0	0	0	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1

V pravdivostní tabulce výše lze vidět, že spojka $\leftrightarrow$ skutečně asociativní je. Oproti tomu spojka $\rightarrow$ asociativní *není* (v pravé části tabulky jsou červeně vyznačeny řádky, kde se hodnota $(\varphi \rightarrow \psi) \rightarrow \chi$ a $\varphi \rightarrow (\psi \rightarrow \chi)$ liší). □

1.3 Terminologie

Nyní zavedeme pojmy které se často používají když mluvíme o formulích výrokové logiky. Ohodnocení proměnných $I \colon \mathbb{X} \to \{0, 1\}$ *splňuje* formuli φ tehdy, když platí, že po dosazení hodnot proměnných v ohodnocení I do formule bude výsledná pravdivostní hodnota formule 1. Jinými slovy, pokud sestrojíme pravdivostní tabulku pro φ a na řádku odpovídajícímu ohodnocení proměnných I bude v posledním sloupci tabulky hodnota 1. V takovém případě říkáme, že I je *modelem* formule φ, což značíme jako $I \models \varphi$ („I *splňuje* φ“). Opačnou vlastnost („I *nesplňuje* φ“) značíme $I \not\models \varphi$. Pro to, abychom zjistili, zda I je či není modelem φ, ovšem není potřeba sestrojit celou pravdivostní tabulku pro φ, ale stačí ji vytvořit jen pro řádek odpovídající I.

Příklad 1.5
Určíme, zda ohodnocení proměnných $I_1 = \{x \mapsto 1, y \mapsto 0, z \mapsto 1\}$ splňuje formuli φ: $(x \wedge (y \to z)) \leftrightarrow (x \vee \neg z)$. Budeme postupovat tak, že si sestrojíme řádek pro I_1 z pravdivostní tabulky pro φ:

x	y	z	$y \to z$	$x \wedge (y \to z)$	$\neg z$	$x \vee \neg z$	φ
1	0	1	1	1	0	1	1

Z tabulky plyne, že I_1 je modelem formule φ, tedy $I_1 \models \varphi$.

Existuje-li nějaké ohodnocení proměnných I takové, že $I \models \varphi$, pak říkáme, že formule φ je *splnitelná*. Z pravdivostní tabulky poznáme splnitelnou formuli jednoduše tak, že se v posledním sloupci vyskytuje alespoň jednou hodnota 1. Formule φ je *nesplnitelná* (neboli *kontradikce*), pokud není splnitelná, tj., v posledním sloupci pravdivostní tabulky pro φ jsou samé hodnoty 0. Formule φ je *platná* (neboli *tautologie*) pokud je splněna libovolným ohodnocením proměnných, což zapisujeme jako $\models \varphi$. Pomocí pravdivostní tabulky můžeme platnou formuli poznat tak, že v posledním sloupci tabulky jsou samé hodnoty 1. φ je *neplatná* (značeno $\not\models \varphi$) pokud existuje ohodnocení proměnných, které je nesplňuje. Pomocí pravdivostní tabulky bychom takovou formuli poznali tak, že by v posledním sloupci byla alespoň jedna hodnota 0. Tyto pojmy lze rozšířit na *množiny* formulí tak, že množina formulí je platná (resp. splnitelná) pokud je konjunkce všech formulí v množině platná (resp. splnitelná).

Příklad 1.6
Zkoumejme typy následujících tří formulí:

$$\begin{aligned} \varphi_1 &: \quad (x \wedge y) \to x \\ \varphi_2 &: \quad (x \vee y) \to x \\ \varphi_3 &: \quad (x \vee y) \wedge \neg(x \vee y) \end{aligned}$$

Jejich pravdivostní tabulky vypadají následovně:

x	y	$x \wedge y$	$(x \wedge y) \to x$	$x \vee y$	$(x \vee y) \to x$	$\neg(x \vee y)$	$(x \vee y) \wedge \neg(x \vee y)$
0	0	0	1	0	1	1	0
0	1	0	1	1	0	0	0
1	0	0	1	1	1	0	0
1	1	1	1	1	1	0	0

Z tabulek lze vidět následující:

- φ_1 je *platná* i *splnitelná*
- φ_2 je *neplatná*, ale *splnitelná*
- φ_3 je *neplatná* a *nesplnitelná*

Z předchozího příkladu lze vidět, že množinu všech formulí výrokové logiky Φ_{VL} lze rozdělit do tří tříd:

1. tautologie,
2. splnitelné, ale neplatné formule a
3. kontradikce.

Na tomto místě je vhodné poznamenat, že $\not\models \varphi$ není to samé jako $\models \neg\varphi$. Vizte následující příklad:

Příklad 1.7
Uvažujme formuli φ: $(x \vee y) \to x$. Její pravdivostní tabulka a pravdivostní tabulka pro její negaci je uvedena níže:

x	y	$x \vee y$	$(x \vee y) \to x$	$\neg((x \vee y) \to x)$
0	0	0	1	0
0	1	1	0	1
1	0	1	1	0
1	1	1	1	0

Z tabulky můžeme vidět, že φ není platná (tj. $\not\models \varphi$), ale ani $\neg\varphi$ není platná (tj. $\not\models \neg\varphi$).

Platí ale následující duality.

Věta 1.2. *Platí následující:*

1. *Formule je platná právě tehdy, když její negace je nesplnitelná.*
2. *Formule je splnitelná právě tehdy, když její negace je neplatná.*

Důkaz. Lze dokázat jednoduše analýzou pravdivostních tabulek. □

Dvě formule φ a ψ jsou *(logicky) ekvivalentní*, zapisováno $\varphi \Leftrightarrow \psi$, pokud pro všechna ohodnocení proměnných I platí, že I je modelem φ právě tehdy, když I je modelem ψ. Z pravdivostní tabulky bychom logicky ekvivalentní formule poznali tak, že by na všech řádcích tabulky měly stejné hodnoty.

Příklad 1.8
Ukážeme, že formule φ: $(x \wedge y) \to z$ a formule ψ: $\neg x \vee \neg y \vee z$ jsou logicky ekvivalentní.

x	y	z	$x \wedge y$	$(x \wedge y) \to z$	$\neg x$	$\neg y$	$\neg x \vee \neg y \vee z$
0	0	0	0	1	1	1	1
0	0	1	0	1	1	1	1
0	1	0	0	1	1	0	1
0	1	1	0	1	1	0	1
1	0	0	0	1	0	1	1
1	0	1	0	1	0	1	1
1	1	0	1	0	0	0	0
1	1	1	1	1	0	0	1

Jak vidno, sloupce pro φ a ψ jsou stejné.

Formule ψ je *logickým důsledkem* formule φ, zapisováno $\varphi \Rightarrow \psi$, tehdy, když pro každé ohodnocení proměnných I platí, že je-li I modelem φ, pak je I rovněž modelem ψ.

1.4 Algebraické úpravy formulí

Často je vhodné s formulemi výrokové logiky různým způsobem manipulovat a upravovat je při zachování jejich sémantiky (např. při *minimalizaci* formulí pro účely kompaktnější hardwarové reprezentace nebo při převodu do některé z normálních forem, viz Sekce 1.5). K tomuto slouží *algebraické úpravy*, které nám umožňují s formulemi pracovat jako s výrazy. Pro algebraické úpravy nám slouží logické ekvivalence v následující Větě. Tyto ekvivalence říkají, jakým způsobem můžeme přepsat podformule ve formuli tak, aby byla zachována sémantika.

Věta 1.3. *Nechť x, y a z jsou výrokové formule. Pak platí následující logické ekvivalence:*

$$
\begin{aligned}
x \wedge (y \wedge z) &\Leftrightarrow (x \wedge y) \wedge z && \textit{(\textbf{asociativita})}\\
x \vee (y \vee z) &\Leftrightarrow (x \vee y) \vee z && \textit{(\textbf{asociativita})}\\
x \wedge y &\Leftrightarrow y \wedge x && \textit{(\textbf{komutativita})}\\
x \vee y &\Leftrightarrow y \vee x && \textit{(\textbf{komutativita})}\\
x \wedge (y \vee z) &\Leftrightarrow (x \wedge y) \vee (x \wedge z) && \textit{(\textbf{distributivita})}\\
x \vee (y \wedge z) &\Leftrightarrow (x \vee y) \wedge (x \vee z) && \textit{(\textbf{distributivita})}\\
\neg(x \wedge y) &\Leftrightarrow \neg x \vee \neg y && \textit{(\textbf{De Morganův zákon})}\\
\neg(x \vee y) &\Leftrightarrow \neg x \wedge \neg y && \textit{(\textbf{De Morganův zákon})}\\
x \wedge (x \vee y) &\Leftrightarrow x && \textit{(\textbf{absorpce})}\\
x \vee (x \wedge y) &\Leftrightarrow x && \textit{(\textbf{absorpce})}\\
x &\Leftrightarrow \neg\neg x && \textit{(\textbf{dvojitá negace})}
\end{aligned}
$$

$$
\begin{array}{rclcrcll}
x \vee x & \Leftrightarrow & x & & x \wedge x & \Leftrightarrow & x & \textit{(\textbf{idempotence})} \\
x \vee 0 & \Leftrightarrow & x & & x \wedge 1 & \Leftrightarrow & x & \textit{(\textbf{neutralita})} \\
x \vee 1 & \Leftrightarrow & 1 & & x \wedge 0 & \Leftrightarrow & 0 & \textit{(\textbf{anihilace})} \\
x \wedge \neg x & \Leftrightarrow & 0 & & x \vee \neg x & \Leftrightarrow & 1 & \textit{(\textbf{komplementarita})}
\end{array}
$$

$$
\begin{array}{rcll}
x \to y & \Leftrightarrow & \neg x \vee y & \textit{(\textbf{implikace})} \\
x \to y & \Leftrightarrow & \neg(x \wedge \neg y) & \textit{(\textbf{implikace})} \\
x \leftrightarrow y & \Leftrightarrow & (x \to y) \wedge (y \to x) & \textit{(\textbf{bikondicionál})}
\end{array}
$$

Důkaz. Každá z ekvivalencí lze dokázat jednoduše například pomocí pravdivostní tabulky. □

Příklad 1.9

Pomocí algebraických úprav převedeme formuli $(x \wedge \neg y) \vee \neg z$ do tvaru, ve kterém se vyskytují pouze spojky $\neg$ a $\to$:

$$
\begin{array}{rcll}
(x \wedge \neg y) \vee \neg z & \Leftrightarrow & \neg\neg(x \wedge \neg y) \vee \neg z & \text{⎧ dvojitá negace⎫} \\
& \Leftrightarrow & \neg(x \to y) \vee \neg z & \text{⎧ implikace⎫} \\
& \Leftrightarrow & (x \to y) \to \neg z & \text{⎧ implikace⎫}
\end{array}
$$

Všimněte si, že v prvním kroku jsme použili pravidlo pro dvojitou negaci *zprava doleva*. Díky použití tohoto pravidla jsme pak mohli použít pravidlo pro implikaci.

Další příklady použití algebraických úprav uvidíme v Sekci 1.5.

1.5 Normální formy

Při mnoha aplikacích výrokové logiky v informatice pracujeme s formulemi v některé z *normálních forem*,tj. s formulemi která splňují jistá syntaktická omezení. V této sekci si představíme tři často používané normální formy.

1.5.1 Negační normální forma (NNF)

Negační normální forma (NNF) je ze zde obsažených nomálních forem nejobecnější. Formule je v NNF pokud:

1. obsahuje jen následující logické spojky: $0, 1, \neg, \wedge, \vee$ a
2. negace $\neg$ se vyskytuje jen před proměnnými.

Často se setkáme s pojmem *literál*, který slouží pro označení proměnné nebo její negace. Např., obě následující formule jsou literály:

1. X,
2. $\neg Y$.

Formule v NNF si pak lze představit jako literály spojené konjunkcemi a disjunkcemi.

Příklad 1.10
Následující formule *jsou* v NNF:

1. $x \wedge \neg y$,
2. $(x \vee y) \wedge (\neg z \vee (\neg x \wedge \neg y))$,
3. x.

Následující formule *nejsou* v NNF:

1. $\neg(x \wedge y)$ (negace je před konjunkcí),
2. $x \wedge \neg\neg y$ (negace je před negací),
3. $x \to \neg y$ (obsahuje spojku $\to$).

Libovolná formule lze převést do NNF následujícím algoritmem:

1. Přepíšeme postupně všechny bikondicionály $\leftrightarrow$ ve formuli za implikace pomocí pravidla *(**bikondicionál**)* z Věty 1.3:

$$x \leftrightarrow y \quad \rightsquigarrow \quad (x \to y) \wedge (y \to x)$$

2. Přepíšeme postupně všechny implikace $\to$ ve formuli za negaci a disjunkci pomocí prvního pravidla *(**implikace**)* z Věty 1.3:

$$x \to y \quad \rightsquigarrow \quad \neg x \vee y$$

3. Pomocí De Morganových zákonů postupně přesuneme negaci co nejhlouběji:

$$\begin{aligned} \neg(x \wedge y) &\quad \rightsquigarrow \quad \neg x \vee \neg y \\ \neg(x \vee y) &\quad \rightsquigarrow \quad \neg x \wedge \neg y \end{aligned}$$

4. Kdykoliv to jde, eliminujeme dvojitou negaci:

$$\neg\neg x \quad \rightsquigarrow \quad x$$

Příklad 1.11
Ukážeme si, jak převést formuli $\neg(x \to \neg(x \wedge y))$ do NNF:

$$\begin{aligned} \neg(x \to \neg(x \wedge y)) &\quad \Leftrightarrow \quad \neg(\neg x \vee \neg(x \wedge y)) && \text{⎧ implikace⎫} \\ &\quad \Leftrightarrow \quad x \wedge (x \wedge y) && \text{⎧ De Morgan⎫} \\ &\quad \Leftrightarrow \quad x \wedge y && \text{⎧ asoc. \& idem.⎫} \end{aligned}$$

Podotkněme, že poslední krok není nezbytný, jelikož formule $x \wedge (x \wedge$

y) již je v NNF (byť lze ještě zjednodušit).

1.5.2 Disjunktivní normální forma (DNF)

Formule je v *disjunktivní normální formě* (DNF)[7], pokud má následující tvar:

$$\bigvee_i \bigwedge_j \ell_{i,j} \tag{1.2}$$

kde $\ell_{i,j}$ je literál (tedy buď proměnná, nebo negace proměnné), tedy jde o disjunkci konjunkcí literálů. V případě DNF se konjunkci literálů říká *klauzule* (např. $(x \wedge y \wedge \neg z)$ je klauzule). Následují příklady formulí v DNF:

1. $(x \wedge y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (x \wedge \neg y \wedge z)$,
2. $(x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \neg x_5) \vee (x_2 \wedge \neg x_3 \wedge x_5)$,
3. $(x \wedge y) \vee \neg z$,
4. $x \wedge \neg y$,
5. $x \vee \neg y$,
6. 1,
7. 0.

U prvních dvou formulí je podmínka DNF jasná. U Formule 3 obsahuje druhá klauzule $(\neg z)$ jen jeden literál, lze si tedy představit, že to je jednoprvková konjunkce. Formuli 3 by šlo tedy zapsat jako $(x \wedge y) \vee (\neg z)$. Disjunkce ve Formuli 4 obsahuje jen jednu klauzuli $(x \wedge \neg y)$. Oproti tomu disjunkce ve Formuli 5 obsahuje dvě jednoduché klauzule (x) a $(\neg y)$. Formule by teda šla zapsat jako $(x) \wedge (\neg y)$. Formule 6 je disjunkce obsahující jednu prázdnou konjunkci (neutrální prvek pro konjunkci je 1). Oproti tomu Formule 7 je prázdná disjunkce (neutrální prvek pro disjunkci je 0).

Poznámka 1.4

DNF se používá při návrhu hardware (např. použití programovatelných obvodů typu PAL/PLA vyžaduje převod Booleovské funkce do DNF, syntéza Booleovské funkce do *lookup tabulek* (LUT) na FPGA také provádí převod do DNF), automatickém usuzování (např. v tzv. *automatických theorem proverech* (ATP)), atp.

1.5.3 Konjunktivní normální forma (CNF)

Formule je v *konjunktivní normální formě* (CNF)[8], pokud má následující tvar:

$$\bigwedge_i \bigvee_j \ell_{i,j} \tag{1.3}$$

[7] někdy se také používá označení *sum of products* (SoP)

[8] někdy se také používá označení *product of sums* (PoS)

kde $\ell_{i,j}$ je literál (tedy buď proměnná, nebo negace proměnné), tedy jde o konjunkci disjunkcí literálů. V případě CNF se disjunkci literálů říká *klauzule* (např. $(x \vee y \vee \neg z)$ je klauzule)[9]. Následují příklady formulí v CNF:

[9] pro odlišení klauzulí v DNF a v CNF se klauzulím ve tvaru $(x \wedge y \wedge \neg z)$ někdy říká *konjunktivní klazule* a klauzulím ve tvaru $(x \vee y \vee \neg z)$ se někdy říká *disjunktivní klauzule*

1. $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z) \wedge (x \vee \neg y \vee z)$,
2. $(x_1 \vee x_2 \vee x_3 \vee x_4 \vee \neg x_5) \wedge (x_2 \vee \neg x_3 \vee x_5)$,
3. $(x \vee y) \wedge \neg z$,
4. $x \vee \neg y$,
5. $x \wedge \neg y$,
6. 0,
7. 1.

Zdůvodnění podmínky CNF pro příklady výše je obdobné jako u DNF.

Poznámka 1.5

CNF se používá např. v programech pro řešení sady omezení (tzv. *constraint solvery*), SAT solverech (tj. nástrojích pro řešení problému splnitelnosti výrokových formulí), automatickém usuzování (např. v tzv. *automatických theorem proverech* (ATP) či nástrojích pro řešení splnitelnosti v prvořádových teoriích (tzv. SMT solverech)), atp.

1.6 Převody do DNF a CNF pomocí algebraických úprav

První způsob, jak převést formuli do DNF nebo CNF je pomocí algebraických úprav ze Sekce 1.4. Konkrétně lze pro převod formule φ do DNF použít následující algoritmus:

1. Převedeme formuli φ na formuli φ_{NNF} v NNF pomocí algoritmu ze Sekce 1.5.1.
2. Formuli φ_{NNF} převedeme do tvaru, kde jsou všechny konjunkce pod disjunkcemi pomocí následujícího distributivního zákona:

$$x \wedge (y \vee z) \quad \rightsquigarrow \quad (x \wedge y) \vee (x \wedge z)$$

Převod formule φ do CNF probíhá obdobně, jen je použit duální distributivní zákon:

1. Převedeme formuli φ na formuli φ_{NNF} v NNF pomocí algoritmu ze Sekce 1.5.1.
2. Formuli φ_{NNF} převedeme do tvaru, kde jsou všechny disjunkce pod konjunkcemi pomocí následujícího distributivního zákona:

$$x \vee (y \wedge z) \quad \rightsquigarrow \quad (x \vee y) \wedge (x \vee z)$$

Příklad 1.12
Ukážeme si, jak převést formuli $((x \vee z) \wedge \neg\neg y) \vee (z \to w)$ do DNF a CNF:

1. Převod do DNF:

$$\begin{aligned}
((x \vee z) \wedge \neg\neg y) \vee (z \to w) &\Leftrightarrow ((x \vee z) \wedge y) \vee (z \to w) && \text{⟅ dvojitá negace⟆}\\
&\Leftrightarrow ((x \vee z) \wedge y) \vee (\neg z \vee w) && \text{⟅ implikace (NNF)⟆}\\
&\Leftrightarrow ((x \wedge y) \vee (z \wedge y)) \vee (\neg z \vee w) && \text{⟅ distributivita⟆}\\
&\Leftrightarrow (x \wedge y) \vee (z \wedge y) \vee \neg z \vee w && \text{⟅ asociativita (DNF)⟆}
\end{aligned}$$

   Formuli jsme nejdříve převedli do NNF a poté jednou použili distributivní zákon. Nakonec jsme smazali zbytečné závorky.

2. Převod do CNF:

$$\begin{aligned}
((x \vee z) \wedge \neg\neg y) \vee (z \to w) &\Leftrightarrow ((x \vee z) \wedge y) \vee (z \to w) && \text{⟅ dvojitá negace⟆}\\
&\Leftrightarrow ((x \vee z) \wedge y) \vee (\neg z \vee w) && \text{⟅ implikace (NNF)⟆}\\
&\Leftrightarrow ((x \vee z) \vee (\neg z \vee w)) \wedge (y \vee (\neg z \vee w)) && \text{⟅ distributivita⟆}\\
&\Leftrightarrow (x \vee z \vee \neg z \vee w) \wedge (y \vee \neg z \vee w) && \text{⟅ asociativita (DNF)⟆}\\
&\Leftrightarrow (x \vee 1 \vee w) \wedge (y \vee \neg z \vee w) && \text{⟅ komplementarita⟆}\\
&\Leftrightarrow 1 \wedge (y \vee \neg z \vee w) && \text{⟅ anihilace⟆}\\
&\Leftrightarrow y \vee \neg z \vee w && \text{⟅ neutralita⟆}
\end{aligned}$$

   Formuli jsme nejdříve převedli do NNF stejně jako při převodu do DNF. Poté jsme jednou použili distributivní zákon a po smazání závorek obdrželi formuli $(x \vee z \vee \neg z \vee w) \wedge (y \vee \neg z \vee w)$ v CNF. Pokračovali jsme ale ještě dalšími zjednodušovacími úpravami a dospěli k ekvivalentní formuli $y \vee \neg z \vee w$, která je také v CNF (konjunkce má jen jednu klauzuli).

   Tady si můžeme povšimnout, že výsledná formule je jak v CNF, tak i v DNF. Proč nám při převodu do DNF vyšla tedy jiná formule? Formule získané algebraickým převodem obecně nejsou *kanonické*, tj. výstupem převodu není jedna unikátní formule. V našem příkladu bychom mohli formuli získanou při převodu do DNF v Bodě 1 ještě dále upravovat:

$$\begin{aligned}
(x \wedge y) \vee (z \wedge y) \vee \neg z \vee w &\Leftrightarrow (x \wedge y) \vee ((z \vee \neg z) \wedge (y \vee \neg z)) \vee w && \text{⟅ distributivita⟆}\\
&\Leftrightarrow (x \wedge y) \vee (1 \wedge (y \vee \neg z)) \vee w && \text{⟅ anihilace⟆}\\
&\Leftrightarrow (x \wedge y) \vee (y \vee \neg z) \vee w && \text{⟅ neutralita⟆}\\
&\Leftrightarrow (x \wedge y) \vee y \vee \neg z \vee w && \text{⟅ asociativita⟆}\\
&\Leftrightarrow y \vee \neg z \vee w && \text{⟅ absorpce⟆}
\end{aligned}$$

   Nyní jsme již získali stejnou formuli jako při předchozím převodu.

1.7 Převody do DNF a CNF pomocí pravdivostní tabulky

Je často jednodušší namísto algebraických úprav dělat převody do DNF a CNF pomocí pravdivostní tabulky[10]. Nejdříve si ukážeme, jak pomocí pravdivostní tabulky provést převod formule do DNF.

[10] Toto platí často jen pro formule s malým počtem proměnných. Při velkém počtu proměnných velikost tabulky neúměrně (exponenciálně) roste, zatímco algebraické úpravy mohou stále pracovat s relativně malými formulemi.

Základní myšlenka převodu do DNF je jednoduchá: v pravdivostní tabulce najdeme právě všechna ohodnocení proměnných $I_1, \ldots, I_\ell$ pro které má formule hodnotu 1. Výsledná formule v DNF se pak sestaví jako disjunkce konjunktivních klauzulí, kde každá klauzule odpovídá právě jednomu ohodnocení I_j pro $1 \leq j \leq \ell$ tak, že se sestaví konjunkce literálů odpovídající tomu, jakou I_j přiřazuje proměnným hodnotu (pokud přiřazuje proměnné x hodnotu 1, pak se vezme literál x, pokud přiřazuje proměnné x hodnotu 0, pak se vezme literál $\neg x$).

Formálně lze tuto konstrukci popsat tak, že pro každé I_j se vytvoří konjunktivní klauzule

$$k_j^{DNF} = \left(\bigwedge_{\substack{x \in \mathbb{X} \\ I_j(x)=0}} \neg x \right) \wedge \left(\bigwedge_{\substack{x \in \mathbb{X} \\ I_j(x)=1}} x \right).$$

a výsledná formule φ v DNF pak vznikne disjunkcí všech klazulí k_j^{DNF}:

$$\varphi_{DNF} = \bigvee_{1 \leq j \leq \ell} k_j^{DNF}.$$

Poznamenejme ještě, že pokud je formule nesplnitelná (nemá pro žádné ohodnocení proměnných hodnotu 1), pak po převodu do DNF dostaneme formuli 0.

Převod formule do CNF probíhá duálně: v pravdivostní tabulce najdeme právě všechna ohodnocení proměnných $I_1, \ldots, I_\ell$ pro které má formule hodnotu 0. Výsledná formule v CNF se pak sestaví jako konjunkce disjunktivních klauzulí, kde každá klauzule odpovídá právě jednomu ohodnocení I_j pro $1 \leq j \leq \ell$ tak, že se sestaví disjunkce literálů negující to, jakou I_j přiřazuje proměnným hodnotu (pokud přiřazuje proměnné x hodnotu 1, pak se vezme literál $\neg x$, pokud přiřazuje proměnné x hodnotu 0, pak se vezme literál x).

Formálně lze tuto konstrukci popsat tak, že pro každé I_j se vytvoří disjunktivní klauzule

$$k_j^{CNF}\colon \left(\bigvee_{\substack{x \in \mathbb{X} \\ I_j(x)=0}} x \right) \vee \left(\bigvee_{\substack{x \in \mathbb{X} \\ I_j(x)=1}} \neg x \right).$$

a výsledná formule φ v CNF pak vznikne konjunkcí všech klazulí k_j^{CNF}:

$$\varphi_{CNF}\colon \bigwedge_{1 \leq j \leq \ell} k_j^{CNF}.$$

Pokud je formule platná (má pro všechna ohodnocení proměnných hodnotu 1), pak po převodu do CNF dostaneme formuli 1.

Příklad 1.13

Ukážeme si převod do DNF a CNF na formuli φ: $(x \to y) \leftrightarrow (y \to z)$. Nejdříve si pro formuli sestavíme pravdivostní tabulku:

x	y	z	$x \to y$	$y \to z$	φ
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	1

V tabulce jsme pro předhlednost zeleně vyznačili ohodnocení proměnných, pro které má formule hodnotu 1 a červeně vyznačili ohodnocení proměnných, pro které má formule hodnotu 0.

1. Převod do DNF: Formule je splněna pro následující čtyři ohodnocení proměnných:

   - $I_{000} = \{x \mapsto 0, y \mapsto 0, z \mapsto 0\}$,
   - $I_{001} = \{x \mapsto 0, y \mapsto 0, z \mapsto 1\}$,
   - $I_{011} = \{x \mapsto 0, y \mapsto 1, z \mapsto 1\}$ a
   - $I_{111} = \{x \mapsto 1, y \mapsto 1, z \mapsto 1\}$.

   Z těchto ohodnocení proměnných sestavíme následující čtyři konjunktivní klauzule:

   - k^{DNF}_{000}: $(\neg x \wedge \neg y \wedge \neg z)$,
   - k^{DNF}_{001}: $(\neg x \wedge \neg y \wedge z)$,
   - k^{DNF}_{011}: $(\neg x \wedge y \wedge z)$ a
   - k^{DNF}_{111}: $(x \wedge y \wedge z)$.

   Disjunkce těchto klauzulí pak bude následující formule φ_{DNF}:

   $$(\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge z) \vee (x \wedge y \wedge z)$$

2. Převod do CNF: Formule není splněna pro následující čtyři ohodnocení proměnných:

   - $I_{010} = \{x \mapsto 0, y \mapsto 1, z \mapsto 0\}$,
   - $I_{100} = \{x \mapsto 1, y \mapsto 0, z \mapsto 0\}$,
   - $I_{101} = \{x \mapsto 1, y \mapsto 0, z \mapsto 1\}$ a

- $I_{110} = \{x \mapsto 1, y \mapsto 1, z \mapsto 0\}$.

Z těchto ohodnocení proměnných sestavíme následující čtyři disjunktivní klauzule:

- k_{010}^{CNF}: $(x \vee \neg y \vee z)$,
- k_{100}^{CNF}: $(\neg x \vee y \vee z)$,
- k_{101}^{CNF}: $(\neg x \vee y \vee \neg z)$ a
- k_{110}^{CNF}: $(\neg x \vee \neg y \vee z)$.

Konjunkce těchto klauzulí pak bude následující formule φ_{CNF}:

$$(x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z)$$

1.8 Systémy logických spojek

Kromě logických spojek $0, 1, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$, se kterými jsme pracovali v předchozích kapitolách, se často můžeme setkat s dalšími spojkami. Nejčastější z nich jsou následující:

- $\oplus$: *exkluzivní disjunkce* (často se jí též říká „xor“ (z anglického „exclusive or“) nebo „nonekvivalence“), kterou lze definovat např. jako $x \oplus y \overset{\text{def}}{\iff} \neg(x \leftrightarrow y)$.
- $\downarrow$: *negovaná disjunkce* (často se používá označení „nor“ z anglického „not or“, můžeme se ale také setkat s termínem *Peircova šipka*): $x \downarrow y \overset{\text{def}}{\iff} \neg(x \vee y)$.
- $\uparrow$: *negovaná konjunkce* (často se používá označení „nand“ z anglického „not and“, můžeme se ale také setkat s termínem *Shefferův operátor*): $x \uparrow y \overset{\text{def}}{\iff} \neg(x \wedge y)$.

Pomocí pravdivostní tabulky můžeme tyto spojky definovat následujícím způsobem:

x	y	$x \oplus y$	$x \downarrow y$	$x \uparrow y$
0	0	0	1	1
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

Naskýtá se otázka, zda jsou všechny doposud zavedené logické spojky nezbytné, nebo jestli si bez některých vystačíme. Definujme si pojem *systém logických spojek* S, což je nějaká podmnožina logických spojek, např. $S = \{\rightarrow, \wedge, 1\}$. Budeme používat označení Φ_S pro množinu všech formulí, které lze vytvořit pomocí spojek z S (a proměnných z $\mathbb{X}$ a závorek). Systém spojek S je *úplný*, pokud pro každou formuli výrokové logiky $\varphi \in \Phi_{VL}$ existuje formule $\varphi_S \in \Phi_S$ taková, že $\varphi \Leftrightarrow \varphi_S$. Uveďme si příklady úplných a neúplných systémů spojek:

- **úplné**: $\{\wedge, \neg\}$, $\{\vee, \neg\}$, $\{\rightarrow, \neg\}$, $\{\uparrow\}$, $\{\downarrow\}$, $\{\oplus, 1\}$, $\{\rightarrow, 0\}$ (a jejich nadmnožiny),
- **neúplné**: $\{\wedge, \vee, \rightarrow, \leftrightarrow, 1\}$ (a libovolná podmnožina).

Příklad 1.14

Dokážeme, že systém spojek $\{\wedge, \neg\}$ je úplný. Toto můžeme dokázat tak, že ukážeme, jak můžeme pomocí spojek $\wedge$ a $\neg$ vyjádřit zbytek spojek:

$$\begin{aligned} x \vee y &\overset{\text{def}}{\iff} \neg(\neg x \wedge \neg y) \\ 0 &\overset{\text{def}}{\iff} x \wedge \neg x \\ 1 &\overset{\text{def}}{\iff} x \vee \neg x \\ x \rightarrow y &\overset{\text{def}}{\iff} \neg x \vee y \\ x \leftrightarrow y &\overset{\text{def}}{\iff} (x \rightarrow y) \wedge (y \rightarrow x) \end{aligned}$$

Všimněte si, že pro definice jednotlivých spojek v uvažovaném systému jsme použili logické ekvivalence z Věty 1.3.

2
Predikátová logika

Oproti *výrokové logice* nám *predikátová logika* poskytuje mnohem bohatší vyjadřovací prostředky, které se používají nejen v matematice pro zapisování vlastností různých struktur, ale stále více v informatice jako jazyk pro specifikaci systémů, práci s databázemi, analýzu a verifikaci softwaru a hardwaru, umělé inteligenci atp. Ve výrokové logice jsme pracovali s formulemi nad výrokovými proměnnými, které reprezentovaly tzv. „prvotní výroky," do nichž jsme se již nedívali. V predikátové logice nás obsah těchto výroků zabývat bude.

V tomto textu se zaměříme na *predikátovou logiku 1. řádu*[1] (v angličtině se často označuje jen termínem *first-order logic (FOL)*). Oproti výrokové logice nám predikátová logika umožňuje mluvit o *entitách* z nějakého *univerza* a jejich *vlastnostech* a *vztazích* mezi nimi (např. o tom, že některá přirozená čísla jsou menší než jiná přirozená čísla).

[1] Existuje i predikátová logika 2. řádu, 3. řádu, atd., které mají ještě větší vyjadřovací schopnosti; těmito se zabývat nebudeme. Zájemci mohou najít hezký úvod do těchto logik např. v [End09], případně mohou kontaktovat autora tohoto textu.

Příklad 2.1
Než se pustíme do formálního popisu syntaxe a sémantiky predikátové logiky, ukažme si na několika příkladech formuli predikátové logiky, co pomocí ní můžeme vyjádřit.

1. Uvažujme následující sylogismus (*Sylogismus* je druh logického tvrzení určité formy, kde je jeden výrok odvozen z dvou jiných výroků (předpokladů). Sylogismy zavedl Aristotelés ve 4. století před naším letopočtem a sloužily jako základ pro logickou dedukci až do poloviny 19. století, kdy George Boole položil základy symbolické logiky.):

   „Všichni muži jsou smrtelní. Sokrates je muž. Tedy Socrates je smrtelný."

   Toto tvrzení lze v predikátové logice zapsat například takto:

   $$\big(\forall x(man(x) \to mortal(x)) \wedge man(Socrates)\big) \to mortal(Socrates).$$

2. Tvrzení

   „Existuje nekonečně mnoho prvočísel."

lze v predikátové logice zapsat například jako

$$\forall x \exists y \Big(y > x \wedge \forall z \big((z \neq 1 \wedge z \neq y) \rightarrow \forall w (wz \neq y) \big) \Big).$$

3. Tvrzení

   „Relace R je tranzitivní.“

   můžeme v predikátové logice zapsat například následující formulí:

$$\forall x \forall y \forall z \big(\, (R(x,y) \wedge R(y,z)) \rightarrow R(x,z) \, \big)$$

4. Uvažujme tabulky `R[jmeno, id]` a `S[id, vek]` v SQL databázi. Následující SQL dotaz

```
select R.jmeno
from R join S on R.id = S.id
where S.vek = 42
```

   lze vyjádřit např. pomocí této formule predikátové logiky:

$$\exists z (R(x,z) \wedge S(z, 42))$$

   kde hodnoty volné proměnné x, pro které formule platí, odpovídají řádkům ve výsledku dotazu.

5. Velká Fermatova věta [Wik21c] lze zapsat jako

$$\forall n \forall x \forall y \big(n > 2 \quad \rightarrow \quad \forall z (x^n + y^n \neq z^n) \big)$$

2.1 Syntaxe

Syntaxi predikátové logiky lze chápat jako rozšíření syntaxe výrokové logiky.

2.1.1 Abeceda

Začneme definicí abecedy, která sestává z následujících prvků:

1. *logické spojky*: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$,
2. *proměnné*: $x, y, z, \ldots, x_1, x_2, \ldots \in \mathbb{X}$, kde $\mathbb{X}$ je spočetně nekonečná množina proměnných,
3. *kvantifikátory*: $\exists, \forall$,
4. *závorky*: $), ($,
5. **funkční symboly**: $f_1, f_2, \ldots \in \mathcal{F}$,
6. **predikátové symboly**: $p_1, p_2, \ldots \in \mathcal{P}$ a
7. *predikátový symbol rovnosti*: $=_{/2}$.

Ve výčtu výše se objevily některé nové termíny, které si teď blíže po-

píšeme. Tučně vyznačené jsou pojmy **funkční symboly** (z množiny $\mathcal{F}$) a **predikátové symboly** (z množiny $\mathcal{P}$). Funkční a predikátové množiny nejsou pevně zafixovány, ale lze je chápat jako „*parametr*“ jazyka, který si volíme podle toho, co chceme v logice vyjádřit. Každý funkční i predikátový symbol má dánu tzv. *aritu*, která říká, kolik parametrů (operandů) daný symbol bere (např. symbol f který bychom používali ve výrazech ve tvaru $f(a, 42, \pi)$ má aritu 3). Aritu lze chápat jako funkci $(\mathcal{F} \cup \mathcal{P}) \to \mathbb{N}$. Má-li např. funkční symbol f aritu n, budeme to značit jako $f_{/n}$.

Uveďme si nějaké příklady používaných funkčních a predikátových symbolů včetně jejich arity:

- *funkční symboly*:
  - $+_{/2}$: binární funkční symbol používaný např. pro operaci *sčítání* (např. $x + (y + 42)$),
  - $\triangle_{/2}$: binární funkční symbol používaný např. pro operaci *symetrického rozdílu množin* (např. $A \triangle B$),
  - $\sin_{/1}$: unární funkční symbol používaný pro operaci *sinus* (např. $\sin(2 \cdot x)$),
  - $e_{/0}$: nulární funkční symbol (tj. symbol pro konstantu) používaný pro *Eulerovo číslo* $2.71828\ldots$ (např. $e + x$),
  - $\texttt{strcpy}_{/2}$, $\texttt{strncpy}_{/3}$, $\texttt{strlen}_{/1}$: funkční symboly odpovídající deklaracím funkcí v hlavičkovém souboru `string.h` standardní knihovny jazyka C (např. $\texttt{strncpy}(y, x, 42)$),
- *predikátové symboly*:
  - $<_{/2}$: binární predikátový symbol používaný např. pro vyjádření vztahu *ostře menší* (např. $x < 5 + y$),
  - $|_{/2}$: binární predikátový symbol používaný např. pro vyjádření vztahu *dělí* (např. $42|x$),
  - $\texttt{isnan}_{/1}$: unární predikátový symbol pro odpovídající makro v hlavičkovém souboru `math.h` standardní knihovny jazyka C vracející hodnotu typu `bool`, které se používá pro určení, zda hodnota výrazu typu `float` je nebo není číslo („*not-a-number: NaN*“) (např. $\texttt{isnan}(\frac{0}{0})$),
  - $S_{/3}$: kde S označuje ternární relaci (použití například pro popis řádku v tabulce S v databázi: $S($`"Chewbacca"`, `"Wookiee"`, `"Kashyyyk"`$)$).

Signatura jazyka predikátové logiky je dána jako dvojice $\langle \mathcal{F}, \mathcal{P} \rangle$. Signaturu můžeme chápat jako *„parametr“* jazyka predikátové logiky[2], až po dodání signatury můžeme začít tvořit samotné formule predikátové logiky. Uvědomme si, že prozatím funkční a predikátové symboly nemají žádný význam; ten jim bude přiřazen až později.

[2] Pro znalce C++ si lze signaturu představit např. jako parametr šablony třídy—až po její instanciaci dostaneme samotnou třídu.

Příklad 2.2
Uveďme si nějaké příklady jazyků predikátové logiky a formulí v těchto jazycích:

1. Jazyk teorie uspořádání se signaturou $\langle \mathcal{F} = \emptyset, \mathcal{P} = \{\leq_{/2}\}\rangle$:
   - $\forall x(x \leq x)$
   - $\forall x \forall y((x \leq y \wedge y \leq x) \rightarrow x = y)$
   - $\forall x \forall y \forall z((x \leq y \wedge y \leq z) \rightarrow x \leq z)$
   - $\forall x \forall y(x \leq y \vee y \leq x)$
2. Jazyk teorie grup se signaturou $\langle \mathcal{F} = \{\cdot_{/2}, e_{/0}\}, \mathcal{P} = \emptyset\rangle$:
   - $\forall x(x \cdot e = x \wedge e \cdot x = x)$
   - $\forall x \exists y(x \cdot y = e \wedge y \cdot x = e)$
3. Jazyk teorie množin se signaturou $\langle \mathcal{F} = \emptyset, \mathcal{P} = \{\in_{/2}\}\rangle$:
   - $\forall u(u \in x \rightarrow u \in y)$
   - $\forall x \exists y \forall z(\forall u(u \in z \rightarrow u \in x) \rightarrow z \in y)$
4. Jazyk teorie polí se signaturou $\langle \mathcal{F} = \{read_{/2}, write_{/3}\}, \mathcal{P} = \emptyset\rangle$:
   - $\forall x \forall y(\forall i(read(x, i) = read(y, i)) \rightarrow x = y)$
5. Jazyk elementární (tzv. Peanovy) aritmetiky se signaturou $\langle \mathcal{F} = \{0_{/0}, S_{/1}, +_{/2}, \cdot_{/2}\}, \mathcal{P} = \emptyset\rangle$:
   - $\forall x \forall y \exists z(x + y = z)$
   - $\forall x \forall y(x \cdot S(y) = x \cdot y + x)$
   - $\exists x \forall y(\neg(x = S(y)))$

V některých příkladech výše jsme použili predikátový symbol $=_{/2}$, který nemusí být v signatuře, protože je v *každém* jazyce predikátové logiky (viz Bod 7 definice abecedy).

2.1.2 Gramatika

Při definici gramatiky predikátové logiky začneme definicí *termu*[3]. Termy nám budou popisovat, jakým způsobem se spočítá nějaká *hodnota*. Například v jazyce s množinou funkčních symbolů $\{+_{/2}, \cdot_{/2}, 3_{/0}\}$ jsou následující zápisy termy:

- „x“,
- „3“,
- „$(x \cdot y) + (x \cdot (y \cdot 3))$“.

Formálně lze *termy* definovat následujícím způsobem:

1. Je-li x proměnná, tj. $x \in \mathbb{X}$, pak řetězec „x“ je term.
2. Je-li f funkční symbol s aritou n a $t_1, \ldots, t_n$ jsou termy, pak i řetězec „$f(t_1, \ldots, t_n)$“ je term.

[3] *Term* je jen jiné označení pro *výraz*, v logice se však téměř výhradně používá slovo term.

Často se používá následující zjednodušení notace:

1. pro symboly binárních operátorů se používá infixový zápis (tj. např. místo „$+(x, 42)$“ píšeme nám dobře známé „$x + 42$“),
2. pro symboly unárních operátorů se používá prefixový nebo postfixový zápis (např. místo „$-(x)$“ se píše „$-x$“ a místo „$^{-1}(x)$“ se píše „x^{-1}“),
3. u konstantních symbolů (tj. symbolů s aritou 0) se nepíšou závorky (např. místo „$\pi()$“ se píše jen „π“).

Příklad 2.3
Uvažujme následující jazyky a příklady termů v nich:

1. Jazyk teorie grup se signaturou $\langle \mathcal{F} = \{\cdot_{/2}, e_{/0}\}, \mathcal{P} = \emptyset\rangle$. Příklady termů v tomto jazyce jsou následující:
   - „$(z \cdot (x \cdot e)) \cdot y$“,
   - „e“,
   - „x“ a
   - „$e \cdot e$“.
2. Jazyk teorie polí se signaturou $\langle \mathcal{F} = \{read_{/2}, write_{/3}\}, \mathcal{P} = \emptyset\rangle$ Příklady termů v tomto jazyce jsou následující:
   - „$read(x, y)$“,
   - „x“ a
   - „$read(write(x, y, z), y)$“.
3. Jazyk elementární (tzv. Peanovy) aritmetiky se signaturou $\langle \mathcal{F} = \{0_{/0}, S_{/1}, +_{/2}, \cdot_{/2}\}, \mathcal{P} = \emptyset\rangle$. Příklady termů v tomto jazyce jsou následující:
   - „$x + y$“,
   - „$x \cdot S(y + S(0 \cdot x))$“ a
   - „x“.

Nyní již můžeme definovat *formule* predikátové logiky:

1. Je-li p predikátový symbol s aritou n a $t_1, \ldots, t_n$ jsou termy, potom je řetězec „$p(t_1, \ldots, t_n)$“ formule (toto platí i pro „vestavěný“ binární predikátový symbol $=$). Formuli tohoto tvaru říkáme *atomická formule*.
2. Jsou-li φ a ψ formule, pak jsou formule i řetězce „$(\neg\varphi)$“, „$(\varphi \wedge \psi)$“, „$(\varphi \vee \psi)$“, „$(\varphi \to \psi)$“ a „$(\varphi \leftrightarrow \psi)$“.
3. Je-li φ formule a $x \in \mathbb{X}$ proměnná, pak jsou formule i řetězce „$(\exists x\varphi)$“ a „$(\forall x\varphi)$“.

Poznámka 2.1

Definice gramatiky pro formuli predikátové logiky φ pomocí Backus-Naurovy formy (BNF) by vypadala následujícím způsobem:

$$\begin{aligned} t &::= x \mid f(t_1, \ldots, t_n) \\ \varphi &::= p(t_1, \ldots, t_m) \mid t_1 = t_2 \mid \\ &\quad (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid (\varphi \leftrightarrow \varphi) \mid \\ &\quad (\forall x \varphi) \mid (\exists x \varphi) \end{aligned}$$

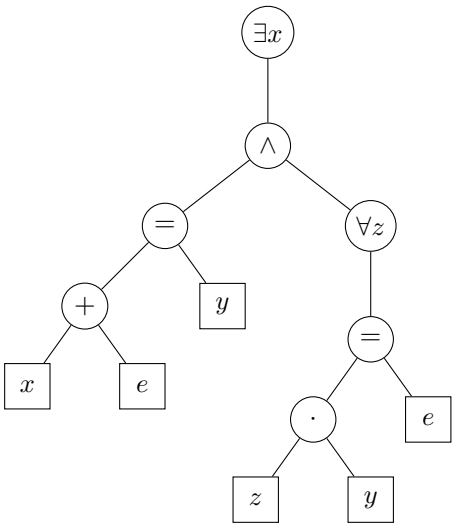
pro $x \in \mathbb{X}$, $f_{/n} \in \mathcal{F}$ a $p_{/m} \in \mathcal{P}$.

Příklady formulí v různých jazycích můžeme vidět v Příkladu 2.2.

Formule predikátové logiky se často (minimálně v programech) reprezentují pomocí jejich syntaktických stromů.

Příklad 2.4

Syntaktický strom formule $\exists x(z + e = y \wedge \forall z(z \cdot y = e))$ může vypadat následujícím způsobem:



2.1.3 Volné a vázané výskyty proměnných

Kvantifikátory hrají ve formulích predikátové logiky velkou roli: vážou proměnnou a umožňují nám mluvit o všech prvcích z univerza diskurzu. Jak uvidíme později v Sekci 2.2.1, formule mluví o tom, co musí platit pro proměnné, které nejsou ve formuli kvantifikovány, tzv. volné proměnné.

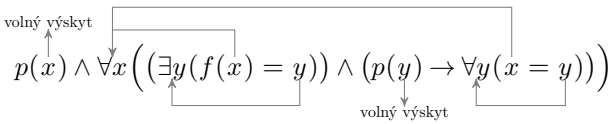
Uvažme následující formuli:

$$\varphi\colon p(\overset{\text{1. výskyt } x}{x}) \wedge \forall x \Big(\big(\exists y(f(\overset{\text{2. výskyt } x}{x}) = \overset{\text{1. výskyt } y}{y})\big) \wedge \big(p(\overset{\text{2. výskyt } y}{y}) \to \forall y(\overset{\text{3. výskyt } x}{x} = \overset{\text{3. výskyt } y}{y})\big)\Big)$$

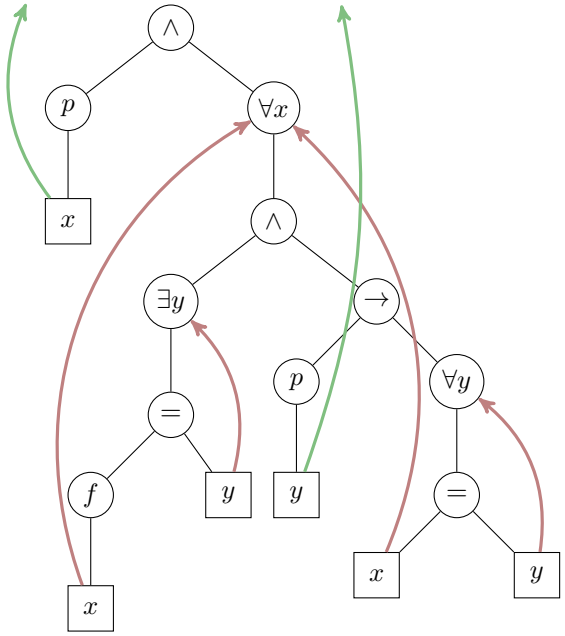
s vyznačenými výskyty proměnných. Výskyt proměnné z ve formuli je *vázaný*, pokud se nachází v oboru platnosti kvantifikátoru $\exists z$ nebo $\forall z$ (ve formuli $\forall z \psi$ je oborem platnosti kvantifikátoru $\forall z$ formule ψ — v syntaktickém stromě formule by to byl podstrom nacházející se pod uzlem $\forall z$). V uvažované formuli jsou obory platnosti jednotlivých kvantifikátorů vyznačeny v následujícím zápisu:

$$p(x) \wedge \forall x \Big(\big(\exists y (f(x) = y)\big) \wedge \big(p(y) \to \forall y (x = y)\big) \Big)$$

Pokud je výskyt proměnné vázaný, pak je vázaný nejbližším kvantifikátorem nad sebou. Pokud výskyt proměnné není vázaný žádným kvantifikátorem, pak je *volný*. V následujícím zápise můžeme vidět, kterými kvantifikátory jsou vázány různé výskyty preměnných x a y, a které výskyty jsou volné:



Syntaktický strom formule s vyznačenými volnými a výzanými výskyty proměnných vypadá takto:



Říkáme, že proměnná je ve formuli *volná*, pokud v ní má alespoň jeden volný výskyt. Formuli φ s volnými proměnnými $x_1, \ldots, x_n$ často značíme jako $\varphi(x_1, \ldots, x_n)$ a používáme notaci $\texttt{FREE}[\varphi] \stackrel{\text{def}}{=} \{x_1, \ldots, x_n\}$. Formuli s volnými proměnnými říkáme *výroková forma*, formuli bez volných proměnných říkáme *uzavřená formule* nebo také *výrok*.

2.2 Sémantika

Sémantika predikátové logiky je podstatně komplikovanější než u výrokové logiky. Zatímco u výrokové logiky nám stačilo pracovat s ohodnocením proměnných přiřazujícím každé proměnné hodnotu 0 nebo 1, v predikátové logice musíme proměnným přiřazovat hodnoty z nějakého univerza a musíme interpretovat funkční a predikátové symboly. Tomuto zevšeobecnění ohodnocení proměnných se říká *realizace* jazyka (někdy též *interpretace* jazyka). Například, máme-li následující formuli:

$$\forall x(f(y) < x),$$

tak abychom určili, zda tato formule platí, nestačí nám jen znát,

1. jakou hodnotu bude mít proměnná y,

ale také to,

2. jaké všechny prvky bude uvažovat kvantifikátor „$\forall x$“,
3. jaká je sémantika funkčního symbolu „$f_{/1}$“ a
4. jaká je sémantika predikátového symbolu „$<_{/2}$“.

Body 1.–4. nám upřesní právě realizace.

Realizace jazyka predikátové logiky se signaturou $\langle \mathcal{F}, \mathcal{P} \rangle$ je dvojice (D_I, α_I) kde:

- D_I je neprázdná množina zvaná *doména* nebo také *univerzum diskurzu* (zdůrazněme nezbytnou podmínku, že D_I musí být *neprázdná*),
- α_I přiřazuje funkčním a predikátovým symbolům jazyka a proměnným sémantiku následujícím způsobem:
  - každému funkčnímu symbolu $f_{/n} \in \mathcal{F}$ přiřazuje n-ární (totální) funkci
    $$f_I \colon \overbrace{D_I \times \cdots \times D_I}^{n} \to D_I,$$
  - každému predikátovému symbolu $p_{/m} \in \mathcal{P}$ přiřazuje m-ární relaci
    $$p_I \subseteq \overbrace{D_I \times \cdots \times D_I}^{m},$$
  - každé proměnné $x \in \mathbb{X}$ přiřazuje hodnotu z domény D_I.

Místo $\alpha_I(f)$, $\alpha_I(p)$ a $\alpha_I(x)$ budeme často psát jen $I(f)$, $I(p)$ a $I(x)$. Části I která určuje doménu a interpretaci symbolů z $\mathcal{F}$ a $\mathcal{P}$ se někdy říká *struktura* a části I která určuje hodnoty proměnných $x \in \mathbb{X}$ se někdy říká *ohodnocení proměnných*.

Příklad 2.5

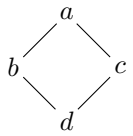
Uvažme jazyk predikátové logiky se signaturou $\langle \mathcal{F} = \{+_{/2}\}, \mathcal{P} = \emptyset \rangle$. Následují příklady realizací tohoto jazyka:

1. Realizace I_1 modelující sčítání přirozených čísel, kde
   - $D_{I_1} = \mathbb{N}$ a
   - $I_1(+) = +_{\mathbb{N}}$ (tj. sčítání přirozených čísel: $+_{\mathbb{N}} = \{(0,0) \mapsto 0, (0,1) \mapsto 1, (1,0) \mapsto 1, (0,2) \mapsto 2, \ldots\}$).
2. Realizace I_2 modelující sčítání tříprvkových vektorů reálných čísel, kde
   - $D_{I_2} = \mathbb{R}^3$ a
   - $I_2(+) = \{([x_1, y_1, z_1], [x_2, y_2, z_2]) \mapsto [x_1 +_{\mathbb{R}} x_2, y_1 +_{\mathbb{R}} y_2, z_1 +_{\mathbb{R}} z_2] \mid x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbb{R}\}$, kde $+_{\mathbb{R}}$ je sčítání reálných čísel.
3. Realizace I_3 modelující spojení (supremum) v Booleově algebře nad $\{0, 1\}$.
   - $D_{I_3} = \{0, 1\}$ a
   - $I_3(+) = \{(0,0) \mapsto 0, (0,1) \mapsto 1, (1,0) \mapsto 1, (1,1) \mapsto 1\}$.
4. Realizace I_4 modelující konkatenaci řetězců v jazyce Python, kde
   - $D_{I_4} =$ `str` (datový typ pro řetězec v jazyce Python) a
   - $I_4(+) = \{(\texttt{"ab"}, \texttt{"cd"}) \mapsto \texttt{"abcd"}, (\texttt{"foo"}, \texttt{"bar"}) \mapsto \texttt{"foobar"}, \ldots\}$.

Pozn. v realizacích výše neuvádíme ohodnocení proměnných; to dává smysl uvádět až v případě, kdy mluvíme o formuli s volnými proměnnými.

Dále uvažujme jazyk predikátové logiky jehož signatura je tato: $\langle \mathcal{F} = \{+_{/2}, \cdot_{/2}, -_{/1}\}, \mathcal{P} = \{E_{/1}\} \rangle$. Následují příklady realizací tohoto jazyka:

5. Realizace I_5 modelující aritmetické operace nad celými čísly s predikátovým symbolem E interpretovaným jako množina sudých čísel:
   - $D_{I_5} = \mathbb{Z}$,
   - $I_5(+) = +_{\mathbb{Z}}$ (tj. sčítání celých čísel),
   - $I_5(\cdot) = \cdot_{\mathbb{Z}}$ (tj. násobení celých čísel),
   - $I_5(-) = \{x \mapsto (0 -_{\mathbb{Z}} x) \mid x \in \mathbb{Z}\}$ (tj. opačné číslo) a
   - $I_5(E) = \{\ldots, -4, -2, 0, 2, 4, \ldots\}$ (tj. sudá čísla).
6. Realizace I_6 modelující svaz na množině $\{a, b, c, d\}$ daný následujícím Hasseovým diagramem



kde

- $D_{I_6} = \{a, b, c, d\}$,
- interpretace symbolů $+_{/2}, \cdot_{/2}, -_{/1}$ jsou dány těmito tabulkami:

$I_6(+)$	a	b	c	d
a	a	a	a	a
b	a	b	a	b
c	a	a	c	c
d	a	b	c	d

$I_6(\cdot)$	a	b	c	d
a	a	b	c	d
b	b	b	d	d
c	c	d	c	d
d	d	d	d	d

	$I_6(-)$
a	d
b	c
c	b
d	a

všimněte si, že $+$ je interpretováno jako supremum, $\cdot$ je interpretováno jako infimum a unární $-$ je interpretováno jako komplement;

- $I_6(E) = \{d\}$ (tj. infimum svazu).

2.2.1 Sémantika formule v realizaci

Sémantika formule je dána realizací, tedy formule samotná bez realizace nemá sémantiku; to znamená, že obecně nemůžeme říct, jestli formule platí nebo neplatí. Když realizaci I máme, můžeme pravdivostní hodnotu formule v realizaci I zjistit intuitivně tak, že do formule za proměnné, funkční a predikátové symboly dosadíme jejich realizace z I a formuli vyčíslíme. První věc kterou musíme spočítat je hodnota všech termů ve formuli pro danou realizaci.

Hodnota termu v realizaci

V pevně dané realizaci I s doménou D_I má term t konkrétní hodnotu: jeden prvek z domény D_I. Pro hodnotu termu t v realizaci I budeme používat notaci $I(t)$. Proměnné $x \in \mathbb{X}$ a konstantní funkční symboly $c_{/0} \in \mathit{funcs}$ mají svou hodnotu přímo dány v realizaci jako $I(x)$ a $I(c)$. Hodnota komplexnějšího termu $t = f(t_1, \ldots, t_n)$ pro n-ární funkční symbol f se definuje induktivně jako

$$I(f(t_1, \ldots, t_n)) \stackrel{\text{def}}{=} f_I(I(t_1), \ldots, I(t_n)) \qquad \text{pro} \qquad f_I = I(f).$$

Neformálně řečeno, vezme se funkce f_I která interpretuje symbol f v realizaci I a do té se dosadí hodnoty termů $t_1, \ldots, t_n$; výsledkem bude opět nějaká hodnota z domény D_I.

Příklad 2.6

Spočítejme hodnotu následujících termů v různých realizacích z Příkladu 2.5.

1. Term $(x + y) + (z + x)$:
   (a) Uvažujme interpretaci I_1 z Příkladu 2.5 rozšířenou o interpretaci proměnných x, y, z takto: $I_1(x) = 3$, $I_1(y) = 42$, $I_1(z) = 17$. Hodnotu $I_1((x + y) + (z + x))$ pak můžeme spočítat násle-

dujícím způsobem:

$$\begin{aligned} I_1((x+y)+(z+x)) &= I_1(x+y) +_{\mathbb{N}} I_1(z+x) \\ &= (I_1(x) +_{\mathbb{N}} I_1(y)) +_{\mathbb{N}} (I_1(z) +_{\mathbb{N}} I_1(x)) \\ &= (3 +_{\mathbb{N}} 42) +_{\mathbb{N}} (17 +_{\mathbb{N}} 3) \\ &= 45 +_{\mathbb{N}} 20 \\ &= 65 \end{aligned}$$

(b) Uvažujme interpretaci I_2 rozšířenou o interpretaci proměnných x, y, z takto: $I_2(x) = [1{,}02; 3{,}57; 2{,}62]$, $I_2(y) = [0; -3{,}12; -4{,}2]$, $I_2(z) = [-1{,}123; 0{,}35; 2{,}56]$. Hodnotu termu $I_2((x+y)+(z+x))$ můžeme spočítat postupně následujícím způsobem:

$$\begin{aligned} &I_2((x+y)+(z+x)) \\ &\quad = I_2(x+y) +_{\mathbb{R}^3} I_2(z+x) \\ &\quad = (I_2(x) +_{\mathbb{R}^3} I_2(y)) +_{\mathbb{R}^3} (I_2(z) +_{\mathbb{R}^3} I_2(x)) \\ &\quad = ([1{,}02; 3{,}57; 2{,}62] +_{\mathbb{R}^3} [0; -3{,}12; -4{,}2]) +_{\mathbb{R}^3} (I_2(z) +_{\mathbb{R}^3} I_2(x)) \\ &\quad = [1{,}02; 0{,}45; -1{,}58] +_{\mathbb{R}^3} (I_2(z) +_{\mathbb{R}^3} I_2(x)) \\ &\quad = [1{,}02; 0{,}45; -1{,}58] +_{\mathbb{R}^3} ([-1{,}123; 0{,}35; 2{,}56] +_{\mathbb{R}^3} [1{,}02; 3{,}57; 2{,}62]) \\ &\quad = [1{,}02; 0{,}45; -1{,}58] +_{\mathbb{R}^3} [-0{,}103; 3{,}92; 5{,}18] \\ &\quad = [0{,}917; 4{,}37; 3{,}6] \end{aligned}$$

(c) Uvažujme interpretaci I_3 rozšířenou o interpretaci proměnných x, y, z takto: $I_3(x) = 0$, $I_3(y) = 1$, $I_3(z) = 0$. Hodnota termu $I_3((x+y)+(z+x))$ bude

$$\begin{aligned} I_3((x+y)+(z+x)) &= I_3(x+y) +_{I_3} I_3(z+x) \\ &= (I_3(x) +_{I_3} I_3(y)) +_{I_3} (I_3(z) +_{I_3} I_3(x)) \\ &= (0 +_{I_3} 1) +_{I_3} (0 +_{I_3} 0) \\ &= 1 +_{I_3} 0 \\ &= 1 \end{aligned}$$

(d) Uvažujme interpretaci I_4 rozšířenou o interpretaci proměnných x, y, z takto: $I_4(x) =$ `"Hello"`, $I_4(y) =$ `" world. "`, $I_4(z) =$ `"Just say "`. Hodnota termu $I_4((x+y)+(z+x))$ bude (symbol „.“ značí konkatenaci řetězců)

$$\begin{aligned} &I_4((x+y)+(z+x)) \\ &\quad = I_4(x+y) \,.\, I_4(z+x) \\ &\quad = (I_4(x) \,.\, I_4(y)) \,.\, (I_4(z) \,.\, I_4(x)) \\ &\quad = (\texttt{"Hello"} \,.\, \texttt{" world. "}) \,.\, (\texttt{"Just say "} \,.\, \texttt{"Hello"}) \\ &\quad = \texttt{"Hello world. "} \,.\, \texttt{"Just say Hello"} \\ &\quad = \texttt{"Hello world. Just say Hello"} \end{aligned}$$

2. Term $(x + y) \cdot (-z)$

   (a) Uvažujme interpretaci I_5 z Příkladu 2.5 rozšířenou o interpretaci proměnných x, y, z takto: $I_5(x) = -17$, $I_5(y) = 42$, $I_5(z) = 3$. Hodnotu $I_5((x + y) \cdot (-z))$ pak můžeme spočítat následujícím způsobem:

$$\begin{aligned} I_5((x + y) \cdot (-z)) &= I_5(x + y) \cdot_{\mathbb{Z}} I_5(-z) \\ &= (I_5(x) +_{\mathbb{Z}} I_5(y)) \cdot_{\mathbb{Z}} (0 -_{\mathbb{Z}} I_5(z)) \\ &= (-17 +_{\mathbb{Z}} 42) \cdot_{\mathbb{Z}} (0 -_{\mathbb{Z}} 3) \\ &= 25 \cdot_{\mathbb{Z}} -3 \\ &= -75 \end{aligned}$$

   (b) Uvažujme interpretaci I_6 rozšířenou o interpretaci proměnných x, y, z takto: $I_6(x) = b$, $I_6(y) = d$, $I_6(z) = b$. Hodnota $I_6((x + y) \cdot (-z))$ bude následující ($\sqcap$ značí infimum, $\sqcup$ značí supremum a $\overline{w}$ značí komplement prvku w):

$$\begin{aligned} I_6((x + y) \cdot (-z)) &= I_6(x + y) \sqcap I_6(-z) \\ &= (I_6(x) \sqcup I_6(y)) \sqcap \overline{I_6(z)} \\ &= (b \sqcup d) \sqcap \overline{b} \\ &= b \sqcap c \\ &= d \end{aligned}$$

Pravdivostní hodnota formule v realizaci

Konečně se dostáváme k vyhodnocení pravdivostní hodnoty formule v realizaci. *Platnost* formule φ v realizaci I, značeno jako $I \models \varphi$ („φ *platí* v I“ nebo „I je *model* φ“), je definována induktivně následujícím způsobem:

1. Je-li φ atomická formule $p(t_1, \ldots, t_m)$, pro m-ární predikátový symbol $p \in \mathcal{P}$, potom

$$I \models p(t_1, \ldots, t_m) \qquad \text{právě když} \qquad (I(t_1), \ldots, I(t_m)) \in p_I,$$

   kde $p_I = I(p)$ je relace interpretující symbol $p_{/m}$ v I (pokud je symbol p unární, pak je p_I množina).

2. Je-li φ atomická formule $t_1 = t_2$ potom

$$I \models t_1 = t_2 \qquad \text{právě když } I(t_1) \text{ a } I(t_2) \text{ jsou identické prvky.}$$

3. Pravdivostní hodnota pro výrokové spojky je definována očekávaným způsobem takto:

$I \models \neg\varphi$ právě když $I \not\models \varphi$,
$I \models \varphi \wedge \psi$ právě když $I \models \varphi$ a $I \models \psi$,
$I \models \varphi \vee \psi$ právě když $I \models \varphi$ nebo $I \models \psi$,
$I \models \varphi \to \psi$ právě když pokud $I \models \varphi$, pak $I \models \psi$ a
$I \models \varphi \leftrightarrow \psi$ právě když buď $I \models \varphi$ a $I \models \psi$, nebo $I \not\models \varphi$ a $I \not\models \psi$,

kde $I \not\models \varphi$ pokud neplatí $I \models \varphi$.

4. Pravdivostní hodnota existenčně a univerzálně kvantifikovaných formulí v realizaci I je definována tak, že

$I \models \exists x\varphi$ pokud existuje realizace I', která rozšiřuje realizaci I o ohodnocení proměnné x na nějakou hodnotu z D_I takové, že $I' \models \varphi$ a
$I \models \forall x\varphi$ pokud pro libovolnou realizaci I', která rozšiřuje realizaci I o ohodnocení proměnné x na nějakou hodnotu z D_I platí, že $I' \models \varphi$.

Příklad 2.7
Uvažujme jazyk L predikátové logiky se signaturou $\langle \mathcal{F} = \{+_{/2}, -_{/1}\}, \mathcal{P} = \{Z_{/1}\}\rangle$ a jeho realizaci I_L s doménou $D_{I_L} = \{a, b, c\}$ a interpretací symbolů

$I_L(+)$	a	b	c
a	a	b	c
b	b	c	a
c	c	a	b

	$I_L(-)$
a	a
b	c
c	b

a $I_L(Z) = \{a\}$.

Zkusíme určit, zda v realizaci I_L platí formule

$$\varphi\colon \forall x \forall y (Z(x) \to x + y = y).$$

Pro určení platnosti formule si můžeme na pomoc vzít tabulku obdobnou pravdivostní tabulce ve výrokové logice, která pro termy obsahuje jejich hodnoty z domény D_{I_L} a pro formule jejich pravdivostní hodnoty (tato tabulka lze vytvořit jen u realizací, jejichž doména je konečná):

x	y	$Z(x)$	$x+y$	$x+y=y$	$Z(x)\to x+y=y$	$\forall y(Z(x)\to x+y=y)$	$\forall x\forall y(Z(x)\to x+y=y)$
a	a	1	a	1	1		
a	b	1	b	1	1	1	
a	c	1	c	1	1		
b	a	0	b	0	1		
b	b	0	c	0	1	1	1
b	c	0	a	0	1		
c	a	0	c	0	1		
c	b	0	a	0	1	1	
c	c	0	b	0	1		

Podotkněme, že platnost formule v realizaci nejde vždy určit, viz následující poznámku.[4]

[4] Jde o tzv. *nerozhodnutelný* problém, tedy o problém, pro jehož řešení neexistuje algoritmus. Více se o nerozhodnutelnosti dozvíte v předmětu Teoretická informatika (TIN).

Poznámka 2.2

Uvažme následující dvě formule v neformální syntaxi predikátové logiky:

$$\varphi_1 \colon \forall x(x>0 \to \exists n(f_1^n(x)=1))$$
$$\varphi_2 \colon \forall x(x>0 \to \exists n(f_2^n(x)=1))$$

kde f^n značí n-násobnou aplikaci funkce f, tj.

$$f^n(x) = \overbrace{(f\circ\cdots\circ f)}^{n}(x)$$

a funkce f_1, f_2 jsou definovány následujícím způsobem:

$$f_1(x) = \begin{cases} \frac{x}{2} & \text{když } x \text{ je sudé} \\ x+1 & \text{když } x \text{ je liché} \end{cases}$$

$$f_2(x) = \begin{cases} \frac{x}{2} & \text{když } x \text{ je sudé} \\ 3x+1 & \text{když } x \text{ je liché} \end{cases}$$

Nechť $I_{\mathbb{N}}$ je standardní realizace elementární aritmetiky nad přirozenými čísly. Potom lze celkem jednoduše dokázat, že $I_{\mathbb{N}} \models \varphi_1$ (můžete si zkusit za domácí úkol). Platí i $I_{\mathbb{N}} \models \varphi_2$? Toto je od roku 1937 otevřený problém matematiky známý pod názvem *Collatzova domněnka* [Wik21a], který se dosud nikomu nepodařilo vyřešit. Může jít o vhodné téma bakalářské práce.

2.3 Terminologie

Formule φ v jazyce L je *splnitelná*, pokud má nějaký model, tedy pokud existuje nějaká realizace I jazyka L taková, že $I \models \varphi$.[5]

[5] Splnitelnost hraje klíčovou roli mimo jiné v *logickém programování*, kde je program zadán jako množina formulí a interpret programovacího jazyka (např. Prologu) se snaží najít model této množiny formulí.

Příklad 2.8
Formule $\varphi\colon \forall x \forall y (x < y \to \exists z (x < z \wedge z < y))$ je splnitelná. Její modely jou například následující realizace:

1. Realizace I_1 s doménou $D_{I_1} = \mathbb{Q}$ a standardní interpretací predikátového symbolu $<_{/2}$ (jako ostrá nerovnost) nad racionálními čísly.
2. Realizace I_2 s doménou $D_{I_2} = \mathbb{R}^3$ a následující interpretací symbolu $<_{/2}$:

$$[x_1, y_1, z_1] <_{I_2} [x_2, y_2, z_2] \overset{\text{def}}{\iff} \sqrt{x_1^2 + y_1^2 + z_1^2} < \sqrt{x_2^2 + y_2^2 + z_2^2}$$

3. Realizace I_3 s doménou $D_{I_3} = \mathbb{Z}$ a následující interpretací symbolu $<_{/2}$:

$$x <_{I_3} y \overset{\text{def}}{\iff} x <_{\mathbb{Z}} y \wedge x \text{ je sudé } \wedge y \text{ je sudé.}$$

4. Realizace I_4 s doménou $D_{I_4} = \mathbb{N}$ a prázdnou interpretací symbolu $p_{/2}$, tj. $I_4(p) = \emptyset$ (formule platí, protože antecedent v implikaci nebude nikdy platit, tedy celá implikace bude triviálně platit).
5. Realizace I_5 s doménou $D_{I_5} = \mathbb{N}$ a $I_5(<) = \mathbb{N} \times \mathbb{N}$ (tj. všechny dvojice čísel jsou v relaci).
6. Realizace I_6 s doménou $D_{I_6} = \{a\}$ a $I_6(<) = \emptyset$. Toto je model formule s nejmenší velikostí.

Naopak, modely formule nejsou např. tyto realizace:

7. Realizace I_7 s doménou $D_{I_7} = \mathbb{N}$ a standardní interpretací predikátového symbolu $<_{/2}$ nad přirozenými čísly.
8. Realizace I_8 s doménou $D_{I_8} = \{a, b\}$ a $I_8(<) = \{(a, b)\}$. Tato realizace je nejmenší realizace, která není modelem formule.

Formule φ je *(logicky) platná* pokud platí v libovolné realizaci. Jinými slovy, pro všechny realizace daného jazyka I platí $I \models \varphi$. Platnost φ značíme podobně jako ve výrokové logice: $\models \varphi$.

Příklad 2.9
Uvažujme jazyk se signaturou $\langle \mathcal{F} = \{1_{/0}, 2_{/0}, +_{/2}\}, \mathcal{P} = \emptyset \rangle$. Je formule

$$\varphi\colon 1 + 1 = 2$$

logicky platná?

Formule φ logicky platná není. Jaktože formule není logicky platná?

Aby byla formule logicky platná, musí platit v *každé* realizaci daného jazyka. Podívejme se ale např. na realizaci I takovou, že

- $D_I = \{a, b\}$,
- $I(1) = a$,
- $I(2) = b$,
- $I(+) = \{(a, a) \mapsto a, \ldots\}$,

V realizaci I je hodnota termu „$1 + 1$“ rovna $I(1 + 1) = a$ a hodnota termu „2“ je rovna $I(2) = b$. Tyto hodnoty nejsou identické a tudíž I není modelem φ, a proto φ ani není logicky platná formule.

Dvě formule φ a ψ jsou *(logicky) ekvivalentní*, zapisováno $\varphi \Leftrightarrow \psi$, pokud pro všechny realizace I platí, že I je modelem φ právě tehdy, když I je modelem ψ.

Formule ψ je *logickým důsledkem* formule φ, zapisováno $\varphi \Rightarrow \psi$, tehdy, když pro každou realizaci I platí, že je-li I modelem φ, pak je I rovněž modelem ψ.

Poznámka 2.3

Výše uvedené vlastnosti (splnitelnost, platnost, ekvivalence, důsledek) jsou všechny obecně nerozhodnutelné (tedy, neexisujte obecný algoritmus, který by pro libovolnou formuli predikátové logiky uměl určit, zda platí nebo ne). Nerozhodnutelnost problému platnosti formule (tzv. *Entscheidungsproblem* [Wik21b]) dokázali nezávisle na sobě Alonzo Church pomocí výpočetního modelu *λ-kalkulu* v roce 1935 [Chu36] a Alan Turing pomocí výpočetního modelu *Turingova stroje* v roce 1936 [Tur37]. O obou těchto výpočetních modelech se dozvíte ve svém studiu více: λ-kalkul je formální základ *funkcionálních programovacích jazyků* (např. Haskell, OCaml, Scheme) a Turingův stroj je formální základ *imperativních programovacích jazyků*.

2.4 Algebraické úpravy formulí

Podobně jako ve výrokové logice, při práci s formulemi predikátové logiky budeme chtít provádět úpravy formulí zachovávajících ekvivalenci. Při těchto úpravách budeme opět vycházet z logicky ekvivalentních formulí. Mimo logických ekvivalencí z výrokové logiky můžeme ještě využít ekvivalence z následující věty.

Věta 2.1. *Pro formule predikátové logiky φ a ψ platí následující ekvi-*

valence:

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}l}
\forall x\varphi & \Leftrightarrow & \neg\exists x(\neg\varphi) & \exists x\varphi & \Leftrightarrow & \neg\forall x(\neg\varphi) & \\
\forall x\varphi & \Leftrightarrow & \varphi & \exists x\varphi & \Leftrightarrow & \varphi & \textit{pokud } x \notin \texttt{FREE}[\varphi] \\
\forall x(\varphi \vee \psi) & \Leftrightarrow & (\forall x\varphi) \vee \psi & \exists x(\varphi \wedge \psi) & \Leftrightarrow & (\exists x\varphi) \wedge \psi & \textit{pokud } x \notin \texttt{FREE}[\psi] \\
\forall x(\varphi \to \psi) & \Leftrightarrow & (\exists x\varphi) \to \psi & \exists x(\varphi \to \psi) & \Leftrightarrow & (\forall x\varphi) \to \psi & \textit{pokud } x \notin \texttt{FREE}[\psi] \\
\forall x(\varphi \to \psi) & \Leftrightarrow & \varphi \to (\forall x\psi) & \exists x(\varphi \to \psi) & \Leftrightarrow & \varphi \to (\exists x\psi) & \textit{pokud } x \notin \texttt{FREE}[\varphi]
\end{array}$$

$$\begin{array}{lcll}
(\forall x(\varphi(x))) \wedge (\forall y(\psi(y))) & \Leftrightarrow & \forall x(\varphi(x) \wedge \psi(x)) & \textit{pokud } x \notin \texttt{FREE}[\psi] \\
(\exists x(\varphi(x))) \vee (\exists y(\psi(y))) & \Leftrightarrow & \exists x(\varphi(x) \vee \psi(x)) & \textit{pokud } x \notin \texttt{FREE}[\psi]
\end{array}$$

Příklad 2.10
Ukážeme si, jak převést formuli

$$\neg\forall x\exists y\Big(\neg\big((p(x) \wedge r(x)) \to r(y)\big)\Big)$$

do tvaru, ve kterém jsou kvantifikátory co nejhlouběji:

$$\begin{aligned}
&\neg\forall x\exists y\Big(\neg\big((p(x) \wedge r(x)) \to r(y)\big)\Big) \\
&\quad\Leftrightarrow \neg\forall x\Big(\neg\forall y\big((p(x) \wedge r(x)) \to r(y)\big)\Big) \\
&\quad\Leftrightarrow \neg\neg\exists x\forall y\big((p(x) \wedge r(x)) \to r(y)\big) \\
&\quad\Leftrightarrow \exists x\forall y\big((p(x) \wedge r(x)) \to r(y)\big) \\
&\quad\Leftrightarrow \exists x\big((p(x) \wedge r(x)) \to \forall y(r(y))\big) \\
&\quad\Leftrightarrow \big(\forall x(p(x) \wedge r(x))\big) \to \forall y(r(y)) \\
&\quad\Leftrightarrow \Big(\big(\forall x(p(x))\big) \wedge \big(\forall x(r(x))\big)\Big) \to \forall y(r(y))
\end{aligned}$$

Výsledná ekvivalentní formule s kvantifikátory co nejhlouběji je tedy

$$\Big(\big(\forall x(p(x))\big) \wedge \big(\forall x(r(x))\big)\Big) \to \forall y(r(y)).$$

2.5 Prenexní normální forma

Základní normální forma v predikátové logice je tzv. *prenexní normální forma*. Tato forma slouží jako základ pro mnoho další úprav formulí, jako je například *Skolemizace* (viz další sekce).

Formule φ je v *prenexní normální formě* (PNF), pokud začíná prefixem kvantifikátorů, za nímž následuje formule bez kvantifikátorů, tj. má následující tvar:

$$\varphi\colon Q_1x_1Q_2x_2\ldots Q_kx_k(\psi(x_1,\ldots,x_k)),$$

kde $Q_1,\ldots,Q_k \in \{\exists,\forall\}$, $x_1,\ldots x_k \in \mathbb{X}$ a ψ je formule bez kvantifikátorů.

Pro převod formule do PNF můžeme použít algebraické úpravy z výrokové logiky a ze Sekce 2.4.

Příklad 2.11
Upravíme formuli

$$\forall n\big(n > 2 \quad \rightarrow \quad \neg\exists x\exists y\exists z(x^n + y^n = z^n)\big)$$

do PNF:

$$\begin{aligned}
&\forall n\big(n > 2 \quad \rightarrow \quad \neg\exists x\exists y\exists z(x^n + y^n = z^n)\big)\\
&\quad\Leftrightarrow \forall n\big(n > 2 \quad \rightarrow \quad \forall x\neg\exists y\exists z(x^n + y^n = z^n)\big)\\
&\quad\Leftrightarrow \forall n\big(n > 2 \quad \rightarrow \quad \forall x\forall y\neg\exists z(x^n + y^n = z^n)\big)\\
&\quad\Leftrightarrow \forall n\big(n > 2 \quad \rightarrow \quad \forall x\forall y\forall z(\neg(x^n + y^n = z^n))\big)\\
&\quad\Leftrightarrow \forall n\forall x\big(n > 2 \quad \rightarrow \quad \forall y\forall z(\neg(x^n + y^n = z^n))\big)\\
&\quad\Leftrightarrow \forall n\forall x\forall y\big(n > 2 \quad \rightarrow \quad \forall z(\neg(x^n + y^n = z^n))\big)\\
&\quad\Leftrightarrow \forall n\forall x\forall y\forall z\big(n > 2 \quad \rightarrow \quad \neg(x^n + y^n = z^n)\big)
\end{aligned}$$

Výsledná formule v PNF je

$$\forall n\forall x\forall y\forall z\big(n > 2 \quad \rightarrow \quad \neg(x^n + y^n = z^n)\big).$$

Příklad 2.12
Upravíme formuli

$$\forall y\Big(\exists x(P(x,y)) \rightarrow Q(y,z)\Big) \wedge \neg\exists z\Big(\forall x\big(R(x,y) \vee Q(x,y)\big)\Big)$$

do PNF:

$$\begin{aligned}
&\forall y\Big(\exists x(P(x,y)) \rightarrow Q(y,z)\Big) \wedge \neg\exists z\Big(\forall x\big(R(x,y) \vee Q(x,y)\big)\Big)\\
&\quad\Leftrightarrow \forall y\Big(\exists x(P(x,y)) \rightarrow Q(y,z)\Big) \wedge \neg\forall x\big(R(x,y) \vee Q(x,y)\big)\\
&\quad\Leftrightarrow \forall w\Big(\exists x(P(x,w)) \rightarrow Q(w,z)\Big) \wedge \neg\forall u\big(R(u,y) \vee Q(u,y)\big)\\
&\quad\Leftrightarrow \forall w\Big(\exists x(P(x,w)) \rightarrow Q(w,z)\Big) \wedge \exists u\Big(\neg\big(R(u,y) \vee Q(u,y)\big)\Big)\\
&\quad\Leftrightarrow \forall w\Big(\big(\exists x(P(x,w)) \rightarrow Q(w,z)\big) \wedge \exists u\big(\neg(R(u,y) \vee Q(u,y))\big)\Big)\\
&\quad\Leftrightarrow \forall w\exists u\Big(\big(\exists x(P(x,w)) \rightarrow Q(w,z)\big) \wedge \neg\big(R(u,y) \vee Q(u,y)\big)\Big)\\
&\quad\Leftrightarrow \forall w\exists u\Big(\forall x\big(P(x,w) \rightarrow Q(w,z)\big) \wedge \neg\big(R(u,y) \vee Q(u,y)\big)\Big)\\
&\quad\Leftrightarrow \forall w\exists u\forall x\Big(\big(P(x,w) \rightarrow Q(w,z)\big) \wedge \neg\big(R(u,y) \vee Q(u,y)\big)\Big)
\end{aligned}$$

Výsledná formule v PNF je

$$\forall y\exists u\forall x\Big(\big(P(x,y) \rightarrow Q(y,z)\big) \wedge \neg\big(R(u,y) \vee Q(u,y)\big)\Big).$$

2.6 Skolemova normální forma

Další normální forma v predikátové logice je tzv. *Skolemova normální forma*. Tato normální forma je používaná hlavně např. při automatickém dokazování formulí predikátové logiky pomocí tzv. *automatizovaných theorem proverů* (angl. *automated theorem provers*) založených na technikách *rezoluce*, *superpozice*, *paramodulace*, atd.

Formule je ve *Skolemově normální formě* (SNF), pokud je v prenexní normální formě a neobsahuje existenční kvantifikátory $\exists$. Zde je potřeba uvědomit si, že obecně neplatí, že by ke každé formuli predikátové logiky existovala formule v SNF, která by byla s původní formulí *logicky ekvivalentní*. Platí ale, že ke každé formuli φ existuje formule φ' taková, že φ' je *splnitelná* právě když je *splnitelná* formule φ (používá se termín *ekvisplnitelnost*). Procesu získání ekvisplnitelné formule v SNF se říká *Skolemizace*.

Základní myšlenka Skolemizace je jednoduchá. Předpokládejme, že máme následující formuli v PNF:

$$\varphi\colon \forall x_1 \forall x_2 \ldots \forall x_k \exists y(\psi(x_1, \ldots, x_k, y)).$$

Tato formule bude splnitelná právě tehdy, když pro každou k-tici hodnot z domény potenciální realizace I, $(v_1, \ldots, v_k) \in D_I^k$, platí, že existuje hodnota $u \in D_I$ taková, že formule $\psi(x_1, \ldots, x_k, y)$ je splnitelná s ohodnocením proměnných $I(x_1) = v_1$, …, $I(x_k) = v_k$ a $I(y) = u$. Snadno lze nahlédnout, že existence daného u pro každou k-tici $(v_1, \ldots, v_k)$ je ekvivalentní existenci k-ární funkce f_y, která každé k-tici hodnot přiřazuje hodnotu u tak, aby byla formule splněna. Můžeme tedy ve formuli φ odstranit kvantifikátor $\exists y$ a nahradit všechny volné výskyty proměnné y ve formuli ψ za term $f_y(x_1, \ldots, x_k)$. Obdrželi bychom tedy formuli

$$\varphi'\colon \forall x_1 \forall x_2 \ldots \forall x_k(\psi(x_1, \ldots, x_k, f_y(x_1, \ldots, x_k))).$$

Povšimněte si, že jsme do jazyka zavedli nový funkční symbol f_y; tomuto funkčnímu symbolu se říká *Skolemova funkce*. Obsahuje-li formule více existenčních kvantifikátorů, Skolemizace postupně odstraňuje kvantifikátory od nejvnějšího po nejvnitřnější.

Příklad 2.13
Převedeme následující formuli na ekvisplnitelnou formuli ve Skolemově normální formě:

$$\forall y \exists u \forall x \Big(\big(P(x, y) \to Q(y, z)\big) \wedge \neg \big(R(u, y) \vee Q(u, y)\big) \Big).$$

Formule již je v PNF, můžeme tedy rovnou pokračovat Skolemizací:

$$\begin{aligned} &\forall y \exists u \forall x \Big(\big(P(x, y) \to Q(y, z)\big) \wedge \neg \big(R(u, y) \vee Q(u, y)\big) \Big) \\ &\quad \Leftrightarrow \forall y \forall x \Big(\big(P(x, y) \to Q(y, z)\big) \wedge \neg \big(R(f_u(y), y) \vee Q(f_u(y), y)\big) \Big) \end{aligned}$$

Příklad 2.14

Převedeme následující formuli na ekvisplnitelnou formuli ve Skolemově normální formě:

$$\exists v \forall x \exists y \forall u \exists z (v + (x + y) = u + z).$$

Formule opět již je v PNF, můžeme tedy rovnou pokračovat Skolemizací:

$$\begin{aligned}
&\exists v \forall x \exists y \forall u \exists z (v + (x + y) = u + z) \\
&\quad \Leftrightarrow \forall x \exists y \forall u \exists z (f_v + (x + y) = u + z) \\
&\quad \Leftrightarrow \forall x \forall u \exists z (f_v + (x + f_y(x)) = u + z) \\
&\quad \Leftrightarrow \forall x \forall u (f_v + (x + f_y(x)) = u + f(x, u))
\end{aligned}$$

3
Literatura

[Bac59] J. W. Backus. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference. In *Proceedings of the International Conference on Information Processing*, pages 125–132. UNESCO, 1959. Typewritten preprint.

[Chu36] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345, 1936.

[End09] Herbert B. Enderton. Second-order and Higher-order Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2018 edition, 2009. https://plato.stanford.edu/archives/win2018/entries/logic-higher-order/.

[Nau61] Peter Naur. A course of Algol 60 programming. *ALGOL Bull.*, (Sup 9):1–38, January 1961.

[Tur37] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.

[Wik21a] Wikipedia contributors. Collatz conjecture — Wikipedia, the free encyclopedia. 2021. https://en.wikipedia.org/w/index.php?title=Collatz_conjecture.

[Wik21b] Wikipedia contributors. Entscheidungsproblem — Wikipedia, the free encyclopedia. 2021. https://en.wikipedia.org/w/index.php?title=Entscheidungsproblem.

[Wik21c] Wikipedia contributors. Fermat's last theorem — Wikipedia, the free encyclopedia. 2021. https://en.wikipedia.org/w/index.php?title=Fermat%27s_Last_Theorem.

Přeloženo 20. února 2024