Olomoucký informatický korespondenční seminář – Olinx

Matematická logika a strojové myšlení, 2. díl

Radim Bělohlávek

olinx.inf.upol.cz

V tomto díle se budeme věnovat spojkám výrokové logiky. Spojky mají pro logiku a informatiku stejný význam jako sčítání, násobení a další operace a funkce pro matematiku a fyziku.

1 Spojky výrokové logiky

Zastavme se nejdříve u samotného pojmu spojka výrokové logiky. Jak jsme viděli, spojka je dána svým symbolem, např. $\neg$ (negace, „ne“) nebo $\wedge$ (konjunkce, „a“), a svou tabulkou pravdivostních hodnot (tabulka popisuje význam, tedy – jak se v logice říká – sémantiku spojky), tedy v našem případě

p	$\neg p$
0	1
1	0

a

$\wedge$	0	1
0	0	0
1	0	1

.

Spojky „ne“, „jestliže, ... pak ...“, „a“, „nebo“ a „právě když“, které jsme v minulém díle zavedli jako spojky výrokové logiky, jsme zvolili proto, že se často vyskytují v přirozeném jazyce, a jsou proto považovány za základní. Uvědomme si však, že v přirozeném jazyku používáme i jiné spojky. Příkladem je spojka „ani ..., ani ...“, kterou budeme označovat symbolem $\downarrow$. Výrok „ani φ, ani ψ“ je pravdivý, právě když jsou oba výroky φ a ψ nepravdivé.[1] Tabulka této spojky tedy vypadá následovně:

$\downarrow$	0	1
0	1	0
1	0	0

. (1)

Spojku $\downarrow$ bychom tedy také mohli zařadit mezi spojky výrokové logiky. Je zřejmé, že formule $\varphi \downarrow \psi$ je ekvivalentní formuli $\neg(\varphi \vee \psi)$, tj.

$$\varphi \downarrow \psi \;\equiv\; \neg(\varphi \vee \psi).$$

Proto se tato spojka někdy nazývá NOR (negace spojky nebo). Jiný název je Peirceova spojka podle logika Charlese S. Peirce (1839–1914), který tuto spojku studoval a zavedl symbol $\downarrow$. Existují i další binární logické spojky? Existují. Každá binární spojka je popsána svou tabulkou pravdivostních hodnot, binárních logických spojek je tedy tolik, kolik existuje takových tabulek. Těch je zřejmě tolik, kolika způsoby lze v tabulce

	0	1
0	a	b
1	c	d

místo a, b, c a d dát pravdivostní hodnoty 0 a 1. Pro $a = 0$, $b = 0$, $c = 0$ a $d = 1$ dostaneme tabulku spojky $\wedge$ (konjunkce), pro $a = 1$, $b = 0$, $c = 0$ a $d = 0$ dostaneme výše uvedenou tabulku spojky $\downarrow$. Za a můžeme dosadit dvě hodnoty (0 nebo 1), nezávisle na tom za b také dvě hodnoty a to samé pro c a d. Je tedy zřejmé, že celkem existuje $2 \cdot 2 \cdot 2 \cdot 2 = 2^4 = 16$ možností, jak tabulku vyplnit. Existuje tedy 16 binárních logických spojek. Ukazuje je následující tabulka, která uvádí i obvyklé symboly a názvy některých spojek.

p	0	0	1	1	zavedený symbol, název
q	0	1	0	1	spojky
$p \circ_1 q$	0	0	0	0	falsum, vždy nepravda
$p \circ_2 q$	1	0	0	0	$\downarrow$, NOR, Peirceova spojka
$p \circ_3 q$	0	1	0	0	
$p \circ_4 q$	1	1	0	0	
$p \circ_5 q$	0	0	1	0	
$p \circ_6 q$	1	0	1	0	
$p \circ_7 q$	0	1	1	0	XOR, výlučné nebo
$p \circ_8 q$	1	1	1	0	$\uparrow$, $\mid$, NAND, Shefferova spojka
$p \circ_9 q$	0	0	0	1	$\wedge$, konjunkce
$p \circ_{10} q$	1	0	0	1	$\leftrightarrow$, ekvivalence
$p \circ_{11} q$	0	1	0	1	druhá projekce
$p \circ_{12} q$	1	1	0	1	$\rightarrow$, implikace
$p \circ_{13} q$	0	0	1	1	první projekce
$p \circ_{14} q$	1	0	1	1	
$p \circ_{15} q$	0	1	1	1	$\vee$, disjunkce
$p \circ_{16} q$	1	1	1	1	verum, vždy pravda

První dva řádky obsahují 4 možné kombinace hodnot výrokových proměnných p a q, další řádky pak popisují šestnáct logických spojek, které jsou označeny $\circ_1, \ldots, \circ_{16}$. Všimněme si, že spojka $\circ_9$ je konjunkce: Druhý sloupec, který odpovídá situaci, kdy p i q mají hodnotu 0 (tj. $e(p) = 0$ a $e(q) = 0$ pro příslušné pravdivostní ohodnocení), ukazuje, že pravdivostní hodnota $p \circ_9 q$ je 0 (tj. $e(p \circ_9 q) = 0$). Z dalších tří sloupců vidíme, že $p \circ_9 q$ má hodnotu 1 jen v případě že p i q mají hodnotu 1, tedy $\circ_9$ je skutečně konjunkce $\wedge$.

Podívejme se na spojky $\circ_7$ a $\circ_{13}$, tedy spojky, které můžeme popsat obvyklými tabulkami následovně (v řádcích jsou hodnoty p, ve sloupcích hodnoty q):

$\circ_7$	0	1
0	0	1
1	1	0

$\circ_{13}$	0	1
0	0	0
1	1	1

Spojka $\circ_7$ nabývá hodnoty 1, právě když má hodnotu 1 právě jeden z argumentů, jinými slovy, když je pravdivé p nebo q, ale ne obě. Proto se tato spojka nazývá výlučné nebo a značí se XOR (z anglického eXclusive OR). Spojka $\circ_{13}$ žádnou přirozenou interpretaci nemá. Nabývá ale stejné pravdivostní hodnoty jako první argument, a proto se někdy nazývá první projekce.

I v případě unárních logických spojek, tedy spojek s jedním argumentem, platí, že jich je více. Tabulka obecné unární spojky c vypadá takto:

p	$c\,p$
0	a
1	b

Spojek je tedy tolik, kolika způsoby lze do tabulky dosadit hodnoty 0 a 1 za a a b. Máme dvě možnosti pro a a nezávisle

[1] Všimněme si ale následující věci. Označují-li φ a ψ výroky „Prší.“ a „Sněží.“, pak výrok $\varphi \downarrow \psi$ čteme „Ani neprší, ani nesněží.“, nikoli „Ani prší, ani sněží.“

na tom dvě možnosti pro b, proto existují celkem $2 \cdot 2 = 4$ unární spojky. Zobrazuje je následující tabulka.

p	0	1	symbol, název
c_1	0	0	falsum
c_2	1	0	$\neg$, negace
c_3	0	1	projekce
c_4	1	1	verum

Je zřejmé, že c_2 je právě spojka negace.

Kromě unárních a binárních spojek můžeme uvažovat i spojky spojující větší počet výroků, například ternární spojky, tedy spojky se třemi argumenty, a obecně n-ární spojky, tedy spojky s n argumenty. Otázkou kolik existuje ternárních a obecně n-árních logických spojek se zabývá první úkol (na konci dílu).

2 Logické funkce

Následující tabulka popisuje ternární spojku f:

p	q	r	$f(p,q,r)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(2)

Tato tabulka vlastně popisuje funkci (neboli zobrazení), která každé kombinaci tří vstupních hodnot 0 a 1 (kombinace jsou v prvních třech sloupcích) přiřazuje jednu z hodnot 0 a 1 (čtvrtý sloupec). Takové funkce se nazývají logické funkce:

Definice 1. Logická funkce s n argumenty neboli *n-ární logická funkce* (říká se také booleovská funkce) je libovolné zobrazení $f : \{0,1\}^n \to \{0,1\}$, tj. zobrazení přiřazující každé n-tici hodnot z množiny $\{0,1\}$ hodnotu 0 nebo 1.

V předchozím díle jsme pro danou formuli určovali její tabulku pravdivostních hodnot. Určovali jsme tedy logickou funkci reprezentovanou danou formulí. Teď se budeme zabývat obrácenou otázkou. Zadáme logickou funkci a budeme hledat logickou formuli, která danou funkci reprezentuje.

V dalším si ukážeme důležitou skutečnost, totiž že každou n-ární logickou funkci lze reprezentovat formulí, která obsahuje jen spojky negace, konjunkce a disjunkce. Konstrukci takové formule si nyní ukážeme.

Předpokládejme, že je dána tabulka popisující logickou funkci f s n argumenty. Tabulka má tedy $n+1$ sloupců, prvních n sloupců odpovídá n argumentům (sloupce jsou označeny $p_1, \ldots, p_n$), poslední sloupec obsahuje hodnoty funkce f. Tabulka má záhlaví a dalších 2^n řádků, protože existuje 2^n kombinací hodnot 0 a 1, které tvoří argumenty funkce f. Hodnota funkce f odpovídající kombinaci vstupních hodnot v řádku i je zapsána v posledním sloupci řádku i. Příkladem takové tabulky pro $n = 3$ je pochopitelně výše uvedená tabulka (2), ve které jsou ovšem sloupce značeny p, q a r, a ne p_1, p_2 a p_3.

Formuli, která reprezentuje danou logickou funkci, obdržíme z tabulky logické funkce následujícím algoritmem:

1. Po každý řádek i, ve kterém funkce nabývá hodnotu 1 sestrojíme konjunkci příslušnou tomuto řádku. Tato konjunkce má tvar $L_1 \wedge L_2 \wedge \cdots \wedge L_n$[2] a sestrojíme ji následovně: Pokud je ve sloupci p_j řádku i hodnota 1, je L_j rovno p_j, pokud je tam hodnota 0, je L_j rovno $\neg p_j$.
2. Výsledná formule vznikne jako disjunkce výše popsaných konjunkcí.

Tato výsledná formule je tzv. úplnou disjunktivní normální formou: Formule, která je disjunkcí několika konjunkcí, z nichž každá obsahuje pro každý výrokový symbol $p_1, \ldots, p_n$ buď přímo symbol p_i nebo jeho negaci $\neg p_i$, se nazývá *úplná disjunktivní normální forma* (ÚDNF). Tedy například pro symboly p_1, p_2 a p_3 formule

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

je ÚDNF, zatímco

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

ne, protože druhá konjunkce neobsahuje ani p_1 ani $\neg p_1$.

Příklad 1. Ukažme si nyní konkrétní příklad popsané konstrukce. Uvažujme logickou funkci popsanou tabulkou (2). Funkce f nabývá hodnoty 1 v řádcích 1, 2, 6 a 8 (nepočítáme záhlaví). V řádku 1 je ve sloupci p hodnota 0, proto je L_1 rovno $\neg p$. Ve sloupcích q a r jsou také hodnoty 0, proto je $L_2 = \neg q$ a $L_3 = \neg r$. Konjunkce $L_1 \wedge L_2 \wedge L_3$ odpovídající řádku 1 je tedy $\neg p \wedge \neg q \wedge \neg r$. Konjunkce odpovídající řádku 2 je $\neg p \wedge \neg q \wedge r$, protože v tomto řádku mají p, q a r hodnoty 0, 0 a 1. Podobně zjistíme, že konjunkce pro řádky 6 a 8 jsou $p \wedge \neg q \wedge r$ a $p \wedge q \wedge r$. Výsledná formule φ_f je tedy následující ÚDNF, která reprezentuje danou logickou funkci f:

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

Pojem reprezentace logické funkce popisuje přesně následující definice.

Definice 2. Formule φ s výrokovými symboly $p_1, \ldots, p_n$ *reprezentuje logickou funkci* $f : \{0,1\}^n \to \{0,1\}$, pokud pro každou n-tici hodnot $b_1, \ldots, b_n \in \{0,1\}$ platí, že hodnota $f(b_1, \ldots, b_n)$ je právě pravdivostní hodnota formule φ při ohodnocení e, při kterém $e(p_1) = b_1, \ldots, e(p_n) = b_n$.

Volně řečeno to znamená, že dosadíme-li do formule φ za její výrokové symboly $p_1, \ldots, p_n$ vstupní argumenty $b_1, \ldots, b_n$ (což jsou hodnoty 0 a 1) funkce f, bude pravdivostní hodnota formule φ stejná jako hodnota funkce f. To tedy znamená, že pokud sestrojíme tabulku pravdivostních hodnot formule, která reprezentuje logickou funkci popsanou zadanou tabulkou, dostaneme tabulku se stejnými hodnotami jako jsou v původní tabulce. Ověřte, že ÚDNF, kterou jsme sestrojili, skutečně reprezentuje logickou funkci popsanou tabulkou (2).

Nyní si předvedeme důkaz, že uvedený algoritmus funguje správně nejen pro logickou funkci uvažovanou v příkladu 1, ale vždy, kromě případu, kdy daná formule je kontradikce (nabývá všude hodnoty 0).[3] Uvažujme libovolnou

[2] $L_1 \wedge L_2 \wedge \cdots \wedge L_n$ je tzv. úplná elementární konjunkce. Závorky zde není potřeba používat, protože konjunkce je asociativní: $\varphi \wedge (\psi \wedge \chi)$ má stejnou pravdivostní hodnotu jako $(\varphi \wedge \psi) \wedge \chi$.

[3] Pak by algoritmus nevybral žádnou konjunkci, a sestrojil by tedy „prázdnou ÚDNF“.

logickou funkci f s n argumenty, která není kontradikce, její tabulku pravdivostních hodnot a výslednou formuli φ ve tvaru ÚDNF, kterou vytvoří náš algoritmus. Musíme ukázat, že pro libovolnou volbu argumentů $b_1, \ldots, b_n$ (to mohou být 0 a 1) a odpovídající ohodnocení e, pro které je $e(p_1) = b_1$, $\ldots$, $e(p_n) = b_n$ je $f(b_1, \ldots, b_n) = 1$, právě když $e(\varphi) = 1$.

Pokud je $f(b_1, \ldots, b_n) = 1$, pak v řádku tabulky, který odpovídá kombinaci $b_1, \ldots, b_n$, je v posledním sloupci hodnota 1, a tedy formule φ obsahuje příslušnou konjunkci $L_1 \wedge \cdots \wedge L_n$ vytvořenou algoritmem. Je snadné si uvědomit, že tato konjunkce nabývá hodnoty 1: Je-li $b_i = 1$, je $L_i = p_i$, tedy $e(L_i) = e(p_i) = b_i = 1$; je-li $b_i = 0$, je $L_i = \neg p_i$, tedy $e(L_i) = e(\neg p_i) = \neg e(p_i) = \neg b_i = \neg 0 = 1$; $L_1 \wedge \cdots \wedge L_n$ je tedy konjunkcí formulí, z nichž každá je v daném ohodnocení e pravdivá, $L_1 \wedge \cdots \wedge L_n$ je tedy v daném ohodnocení sama pravdivá, tj. $e(L_1 \wedge \cdots \wedge L_n) = 1$. Protože φ je disjunkcí několika konjunkcí a jednou z nich je pravdivá konjunkce $L_1 \wedge \cdots \wedge L_n$, je φ při ohodnocení e pravdivá, tj. $e(\varphi) = 1$.

Pokud je naopak $e(\varphi) = 1$, musí být pravdivá některá z konjunkcí, ze kterých se φ skládá. Nechť je to konjunkce $L_1 \wedge \cdots \wedge L_n$. Podle toho, jak algoritmus konjunkce vytváří, je jasné (podobnou úvahou jako výše), že tato konjunkce byla sestrojena právě na řádku, ve kterém se nachází kombinace $b_1, \ldots, b_n$. V posledním sloupci tohoto řádku musím být hodnota 1 (jinak by tam algoritmus tuto konjunkci nevytvořil), což ale znamená, že $f(b_1, \ldots, b_n) = 1$. Důkaz je hotov.

Místo ÚDNF jsme mohli použít „duální“ tvar formule, ve kterém se zamění role konjunkce a disjunkce: Formule, která je konjunkcí několika disjunkcí, z nichž každá obsahuje pro každý výrokový symbol $p_1, \ldots, p_n$ buď přímo symbol p_i nebo jeho negaci $\neg p_i$, se nazývá *úplná konjunktivní normální forma* (ÚKNF). Pro symboly p_1, p_2 a p_3 je formule

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

ÚKNF. K sestrojení ÚKNF, která reprezentuje logickou funkci zadanou tabulkou, můžeme použít algoritmus podobný výše uvedenému algoritmu pro ÚDNF. Změny oproti výše uvedenému algoritmu jsou tyto:

- procházíme řádky tabulky, ve kterých je v posledním sloupci hodnota 0 (a ne 1);
- pro každý takový řádek sestrojíme disjunkci $L_1 \vee \cdots \vee L_n$ (a ne konjunkci);
- pokud je ve sloupci p_j řádku i hodnota 1, je L_j rovno $\neg p_j$, pokud je tam hodnota 0, je L_j rovno p_j (to je opačně než ve výše uvedeném algoritmu);
- výsledná ÚKNF je konjunkcí takto vytvořených disjunkcí.

Zdůvodněte, proč tento postup vede k vytvoření formule ve tvaru ÚKNF, která reprezentuje danou logickou funkci.

Příklad 2. Je dána formule $(p \to q) \wedge (p \to r)$. Sestrojíme ÚDNF a ÚKNF, které jsou ekvivalentní se zadanou formulí. Nejdříve sestrojíme tabulku dané formule, tj. tabulku popisující logickou funkci, kterou tato formule reprezentuje:

p	q	r	$(p \to q) \wedge (p \to r)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(3)

Projdeme řádky, ve kterých má formule $(p \to q) \wedge (p \to r)$ pravidvostní hodnotu 1, vytvoříme příslušné konjunkce a napíšeme je do sloupce označeného „ÚDNF“. Poté projdeme řádky, ve kterých má formule pravidvostní hodnotu 0, vytvoříme příslušné disjunkce a napíšeme je do sloupce označeného „ÚKNF“. Dostaneme tak následující tabulku.

p	q	r	$(p \to q) \wedge (p \to r)$	ÚDNF	ÚKNF
0	0	0	1	$\neg p \wedge \neg q \wedge \neg r$	
0	0	1	1	$\neg p \wedge \neg q \wedge r$	
0	1	0	1	$\neg p \wedge q \wedge \neg r$	
0	1	1	1	$\neg p \wedge q \wedge r$	
1	0	0	0		$\neg p \vee q \vee r$
1	0	1	0		$\neg p \vee q \vee \neg r$
1	1	0	0		$\neg p \vee \neg q \vee r$
1	1	1	1	$p \wedge q \wedge r$	

Výsledná ÚDNF vznikne disjunkcí z konjunkcí uvedených v příslušném sloupci. Je to tedy formule

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

Podobně výsledná ÚKNF vznikne konjunkcí z disjunkcí uvedených ve sloupci pro ÚKNF, je to tedy formule

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

Příklad tím končí.

To, že ke každé logické funkci (kromě kontradikce) můžeme sestrojit ve výše uvedeném smyslu její ÚDNF a že v ÚDNF se nevyskytují jiné spojky než $\neg$, $\wedge$ a $\vee$ znamená, že systém spojek $\{\neg, \wedge, \vee\}$ stačí pro reprezentaci jakékoli logické funkce. (Kontradikci můžeme reprezentovat například formulí $p \wedge \neg p$.) Takové systémy se nazývají funkčně úplné.

Definice 3. Systém $\{c_1, \ldots, c_k\}$ logických spojek je *funkčně úplný*, jestliže lze každou logickou funkci f s n argumenty reprezentovat nějakou logickou formulí, který obsahuje jen spojky ze systému $\{c_1, \ldots, c_k\}$.

Na konci prvního dílu jsme si ukázali, že

$$p \vee q \equiv \neg(\neg p \wedge \neg q), \tag{4}$$

tedy že formule $p \vee q$ a $\neg(\neg p \wedge \neg q)$ jsou ekvivalentní, což podle definice znamená, že pro každé pravdivostní ohodnocení e je pravdivostní hodnota $e(p \vee q)$ stejná jako pravdivostní hodnota $e(\neg(\neg p \wedge \neg q))$. To je vidět z následující tabulky pravdivostních hodnot:

p	q	$p \vee q$	$\neg(\neg p \wedge \neg q)$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Spojku disjunkce ($\vee$) tedy můžeme definovat pomocí negace a konjunkce ($\neg$ a $\wedge$). Každou formuli, obsahující jen spojky $\neg$, $\wedge$ a $\vee$, tedy můžeme převést na ekvivalentní formuli obsahující jen $\neg$ a $\wedge$: výskyty spojky $\vee$ postupně odstraníme dle (4). Například formuli

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

převedeme (náhradou prvního výskytu $\vee$ zleva) na

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

a nakonec na

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

Vzhledem k tomu, že systém spojek $\{\neg, \wedge, \vee\}$ stačí pro reprezentaci jakékoli logické funkce, má tedy tuto vlastnost i systém $\{\neg, \wedge\}$, který obsahuje jen dvě spojky: K dané logické funkci můžeme sestrojit odpovídající formuli obsahující jen $\neg$, $\wedge$ a $\vee$ (takovou formulí je např. její ÚDNF), tu pak převedeme na formuli obsahující jen $\neg$ a $\wedge$, jak ukazuje právě uvedený příklad.

Podobně platí (ověřte):

$$\begin{aligned}
p \wedge q &\equiv \neg(\neg p \vee \neg q), && \text{tj. } \wedge \text{ lze definovat pomocí } \neg \text{ a } \vee,\\
p \to q &\equiv \neg p \vee q, && \text{tj. } \to \text{ lze definovat pomocí } \neg \text{ a } \vee,\\
p \to q &\equiv \neg(p \wedge \neg q), && \text{tj. } \to \text{ lze definovat pomocí } \neg \text{ a } \wedge,\\
p \vee q &\equiv \neg p \to q, && \text{tj. } \vee \text{ lze definovat pomocí } \neg \text{ a } \to,\\
p \wedge q &\equiv \neg(p \to \neg q), && \text{tj. } \wedge \text{ lze definovat pomocí } \neg \text{ a } \to.
\end{aligned}$$

Protože víme, že systém $\{\neg, \wedge, \vee\}$ je funkčně úplný, je funkčně úplný i systém $\{\neg, \vee\}$: konjunkci $\wedge$ lze totiž vyjádřit pomocí $\neg$ a $\vee$, a tudíž ji nepotřebujeme. Systém $\{\neg, \to\}$ je také funkčně úplný: konjunkci $\wedge$ i disjunkci $\vee$ lze dle toho, co jsme právě uvedli, vyjádřit pomocí $\neg$ a $\to$. Formuli, která obsahuje jen $\neg$, $\wedge$ a $\vee$ můžeme totiž převést na formuli obsahující jen $\neg$ a $\to$. Například formuli

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

převedeme dle výše uvedených ekvivalencí na

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

a pak na

$$\neg(p \to \neg(\neg q \to \neg r)).$$

Platí tedy následující věta:

Věta 1. *Každý z následujících systémů spojek je funkčně úplný, tedy stačí k vyjádření libovolné logické funkce:*

- $\{\neg, \wedge, \vee\}$,
- $\{\neg, \wedge\}$,
- $\{\neg, \vee\}$,
- $\{\neg, \to\}$.

Vzniká otázka, zda existuje systém s pouhou jednou spojkou, který je funkčně úplný. Taková spojka musí být pochopitelně binární, protože z unární spojky lze vytvořit pouze formule s jedním výrokovým symbolem (pro negaci jsou to formule $\neg p$, $\neg\neg p$, $\neg\neg\neg p$ atd.). Spojka $\wedge$ ale nestačí, tj. systém $\{\wedge\}$ není funkčně úplný. To vyplývá z následující úvahy. Spojka $\wedge$ je monotónní, tj. čím větší jsou její argumenty, tím větší je její výsledek (přesněji: když pro pravdivostní hodnoty $a, b, c, d \in \{0, 1\}$ je $a \leq b$ a $c \leq d$, pak je $a \wedge c \leq b \wedge d$). Snadno se ukáže, že skládáním monotónních funkcí může vzniknout jen monotónní funkce. Ale negace není monotónní, neboť $0 \leq 1$, ale $\neg 0 \not\leq \neg 1$. Tedy negaci nelze vyjádřit jen pomocí $\wedge$. Podobně lze ukázat, že ani $\{\vee\}$ ani $\{\wedge\}$ nejsou funkčně úplné.

Existují však dvě binární logické spojky, z nichž každá má tu zajímavou vlastnost, že sama tvoří funkčně úplný systém. První z nich je Peirceova spojka $\downarrow$, kterou jsme si uvedli na začátku tohoto dílu. Je definována tabulkou (1), tedy:

$\downarrow$	0	1
0	1	0
1	0	0

K prokázání, že systém $\{\downarrow\}$ je funkčně úplný, stačí ověřit například to, že jak negaci $\neg$, tak konjunkci $\wedge$ lze vyjádřit pomocí $\downarrow$ (víme totiž, že $\{\neg, \wedge\}$ je sám o sobě funkčně úplný systém). Snadno se ale ověří, že

$$\begin{aligned}
\neg p &\equiv p \downarrow p,\\
p \wedge q &\equiv (p \downarrow p) \downarrow (q \downarrow q).
\end{aligned}$$

Ověřte to: vytvořte tabulky pravdivostních hodnot pro formule $p \downarrow p$ a $(p \downarrow p) \downarrow (q \downarrow q)$.

Druhou binární logickou spojkou, která sama tvoří funkčně úplný systém, je Shefferova spojka $\uparrow$, kterou jsme také zavedli na začátku tohoto dílu. Je dána následující tabulkou:

$\uparrow$	0	1
0	1	1
1	1	0

Ukázat, že $\uparrow$ je funkčně úplná, je předmětem úkolu na konci tohoto dílu.

3 Zkracování logických formulí

Pro člověka není příjemné pracovat s dlouhými formulemi. Jsou nepřehledné a těžko se chápe jejich smysl. V předchozí kapitole jsme například odvodili ÚDNF

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

což je formule, která sice reprezentuje logickou funkci (3), ale není snadné ji přečíst a pochopit její smysl. Upřednostnili bychom formuli ekvivalentní, ale kratší. Je to podobná situace, jako když to, co lze popsat krátkou větou nebo souvětím, popíšeme souvětím, které je zbytečně dlouhé. Při použití logiky v umělé inteligenci je skutečně často potřeba předložit uživateli formuli, která je krátká. Taková formule může například uživateli poskytovat informaci o tom, co je uloženo v datech, a pokud je to formule dlouhá, není užitečná.

Představíme si nyní metodu pro hledání krátké formule, která je ekvivalentní zadané logické formuli. Jde o metodu tzv. *Karnaughovy mapy*, kterou navrhl americký matematik a fyzik Maurice Karnaugh (nar. 1924) [1]. Metodu vysvětlíme pouze pro případ, kdy formule obsahuje právě 4 výrokové symboly, p, q, r a s.

K zadané logické formuli φ nejdříve sestrojíme její tabulku pravdivostních hodnot, tedy popíšeme pravdivostní funkci f, kterou tato formule reprezentuje. Tuto tabulku ale nakreslíme jinak, než jak jsme to dělali doposud. Tabulka, kterou

budeme používat, se nazývá Karnaughova mapa. Příkladem je následující tabulka (mapa).

$rs \backslash pq$	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	1	1	1
10	0	1	0	0

. (5)

Tato mapa popisuje logickou funkci f následovně. První řádek obsahuje možné kombinace hodnot proměnných p a q (první je hodnota p, druhá je hodnota q), tj. 00, 01, 11, 10. Podobně první sloupec obsahuje kombinace hodnot proměnných r a s. Všimněte si, že tyto kombinace v první řádku i v prvním sloupci jsou řazeny tak, že dvě sousední se liší právě v jedné pozici, což je pro metodu důležité.

Hodnoty logické funkce jsou v mapě zapsány přirozeně: Například hodnota $f(0,1,1,0)$, je zapsána na průsečíku sloupce nadepsaného 01 (hodnota p je 0, hodnota q je 1) a řádku nadepsaného 10 (hodnota r je 1, hodnota s je 0). Na tomto místě je 1, což znamená, že $f(0,1,1,0) = 1$. Na průsečíku sloupce 11 a řádku 00 je 0, tedy $f(1,1,0,0) = 0$.

Krátkou formuli, která reprezentuje funkci popsanou Karnaughovou mapou, vytvoříme následovně. Tam, kde jsou v mapě zapsané hodnoty funkce f, se snažíme najít maximální obdélníkové části, které jsou plné jedniček. Přitom počet políček pokrytých obdélníkem musí být mocninou dvojky, tedy 1, 2, 4, 8. Příkladem je následující obdélník, který sestává z políček s tučně napsanou jedničkou.

$rs \backslash pq$	00	01	11	10
00	**1**	**1**	0	0
01	**1**	**1**	0	0
11	0	1	1	1
10	0	1	0	0

.

Další dva obdélníky jsou znázorněny v následujících tabulkách.

$rs \backslash pq$	00	01	11	10
00	1	**1**	0	0
01	1	**1**	0	0
11	0	**1**	1	1
10	0	**1**	0	0

$rs \backslash pq$	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	1	**1**	**1**
10	0	1	0	0

.

Všimněte si, že druhý obdélník se překrývá s prvním. Takové překryvy jsou povolené. Všimněme si také, že všechny tyto obdélníky jsou maximální: žádný není možné zvětšit do šířky, ani do výšky, aniž bychom porušili podmínku, že v obdélníku musí být jen jedničky. Ani třetí obdélník není možné zvětšit: Kdybychom přidali jedničku ve sloupci 01 a řádku 11, získali bychom sice větší obdélník, ale pokrýval by 3 políčka, což nelze, protože 3 není mocninou dvojky.

Snažíme se najít nejmenší počet takových obdélníků, které pokrývají všechny jedničky v tabulce. Naše tři obdélníky to zřejmě splňují. Ke každému obdélníku nyní sestrojíme konjunkci z výrokových symbolů p, q, r a s a negací těchto symbolů. Taková konjunkce nemusí obsahovat každý výrokový symbol, jako tomu bylo při konstrukci ÚDNF. Konstrukce takové konjunkce bude zřejmá z příkladů. První obdélník odpovídá sloupcům 00 a 01 a řádkům 00 a 01. Sloupce odpovídají symbolům p a q. Protože hodnota p ve sloupcích obdélníku se nemění (je 0), zahrneme p do vytvářené konjunkce, ovšem vzhledem tomu, že hodnota p je 0, zahrneme tam $\neg p$. Hodnota q ve sloupcích obdélníku se mění (jednou je 0, jednou 1), proto tam q nezahrneme. Podobně pro řádky: do vytvářené konjunkce zahrneme $\neg r$, protože hodnota r se nemění a je 0; hodnota s se mění, proto s ani $\neg s$ nezahrnujeme. Z prvního obdélníku tedy vytvoříme konjunkci

$$\neg p \wedge \neg r.$$

Z druhého obdélníku vytvoříme konjunkci

$$\neg p \wedge q,$$

protože ve sloupci 01 tohoto obdélníku se hodnoty p a q nemění, zatímco v řádcích tohoto obdélníku (00, 01, 11 a 10) se mění jak hodnota r, tak hodnota s. Z třetího obdélníku vytvoříme konjunkci

$$p \wedge r \wedge s,$$

protože ve sloupcích tohoto obdélníku se nemění hodnota p, tato hodnota je 1 a v řádcích se nemění hodnoty r ani s a tyto hodnoty jsou v obou případech 1.

Tyto konjunkce nakonec spojíme v disjunkci, čímž dostaneme výslednou formuli

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

Dostali jsme tak poměrně krátkou formuli reprezentující logickou funkci 4 argumentů, která je popsána výše uvedenou tabulkou. Uvědomte si, že kdybychom sestrojili ÚDNF této funkce, byla by to disjunkce sestávající z osmi konjunkcí, přitom každá z těchto konjunkcí obsahuje všechny čtyři výrokové symboly. Formule sestrojená Karnaughovou mapou je tedy mnohem kratší.

Úkol 1 *8 bodů*

V kapitole 1 tohoto dílu jsme odvodili, že existují 4 unární logické spojky a 16 binárních spojek. Uvědomme si, že v principu můžeme uvažovat také ternární spojky (tj. spojky se 3 argumenty) a obecně i n-ární spojky. Například tabulka

p	q	r	$c(p,q,r)$	$d(p,q,r)$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

popisuje dvě ternární spojky, c a d. Spojku c lze přitom chápat jako ternární konjunkci, protože nabývá hodnoty 1 právě v případě, že všechny tři argumenty jsou pravdivé. Spojka d se nazývá podmíněná disjunkce a zavedl ji americký logik Alonzo Church. Slovně ji lze popsat takto:

jestliže p, potom q, jinak r.

Lze ji vyjádřit například formulí $(p \to q) \wedge (\neg p \to r)$, popř. formulí $(p \wedge q) \vee (\neg p \wedge r)$.

Otázka: Kolik existuje ternárních logických spojek? Zdůvodněte.

Obecněji můžeme uvažovat n-ární logické spojky, tedy spojky s n argumenty. Část tabulky popisující takovou

spojku c je zde:

p_1	p_2	$\cdots$	p_n	$c(p_1, \ldots, p_n)$
0	0	$\cdots$	0	0
0	0	$\cdots$	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	$\ldots$	1	1

Otázka: Kolik existuje n-árních logických spojek? Zdůvodněte.

Úkol 2 *8 bodů*

- Najděte k formuli $p \leftrightarrow (\neg q \vee r)$ ekvivalentní formuli, která obsahuje jen spojky $\neg$ a $\rightarrow$.
- Najděte k formuli $p \,\mathrm{XOR}\, \neg q$ ekvivalentní formuli, která obsahuje jen spojky $\neg$ a $\wedge$.
- Dokažte podobně jako jsme to udělali pro Peirceovu spojku, že Shefferova spojka $\uparrow$ sama tvoří funkčně úplný systém.

Úkol 3 *9 bodů*

Pomocí Karnaughovy mapy sestrojte krátkou logickou formuli, která reprezentuje následující logickou funkci. Funkce f má čtyři argumenty a nabývá hodnoty 1, právě když počet jejích argumentů, které jsou jedničky, je sudý. Tedy je $f(0,0,0,0) = 1$ (nula je sudé číslo), $f(0,1,0,0) = 0$ (pouze jeden argument je jednička), $f(0,1,0,1) = 1$ (dva argumenty mají hodnotu 1) apod.

Literatura

[1] M. Karnaugh, 1953. The Map Method for Synthesis of Combinational Logic Circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*. 72 (9): 593–599.

Katedra informatiky
Univerzita Palackého v Olomouci
17. listopadu 12
771 46 Olomouc