Kapitola 1

Výroková logika

1.1 Výroky

1.1.1 Výroky. Máme danou neprázdnou množinu A tzv. *atomických výroků* (též jim říkáme *logické proměnné*). Konečnou posloupnost prvků z množiny A, logických spojek a závorek nazýváme *výroková formule* (zkráceně jen *formule*), jestliže vznikla podle následujících pravidel:

1. Každá logická proměnná (atomický výrok) $a \in A$ je výroková formule.

2. Jsou-li α, β výrokové formule, pak $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \Rightarrow \beta)$ a $(\alpha \Leftrightarrow \beta)$ jsou také výrokové formule.

3. Nic jiného než to, co vzniklo pomocí konečně mnoha použití bodů 1 a 2, není výroková formule.

Všechny formule, které vznikly z logických proměnných množiny A, značíme $\mathcal{P}(A)$.

1.1.2 Poznámka. Spojka $\neg$ se nazývá *unární*, protože vytváří novou formuli z jedné formule. Ostatní zde zavedené spojky se nazývají *binární*, protože vytvářejí novou formuli ze dvou formulí.

V dalším textu logické proměnné označujeme malými písmeny např. $a, b, c, \ldots$ nebo $x, y, z, \ldots$, výrokové formule označujeme malými řeckými písmeny např. $\alpha, \beta, \gamma, \ldots$ nebo $\varphi, \psi, \ldots$. Také většinou nebudeme ve formulích psát ty nejvíc vnější závorky — tj. píšeme $a \vee (b \Rightarrow c)$ místo $(a \vee (b \Rightarrow c))$.

1.1.3 Syntaktický strom formule. To, jak formule vznikla podle bodů 1 a 2, si můžeme znázornit na *syntaktickém stromu*, též *derivačním stromu* dané formule. Jedná se o kořenový strom, kde každý vrchol, který není listem je ohodnocen logickou spojkou a jedná-li se o binární spojku, má vrchol dva následníky, jedná-li se o unární spojku, má vrchol pouze jednoho následníka. Přitom pro formule tvaru $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \Rightarrow \beta)$ odpovídá levý následník formuli α, pravý následkník formuli β. Listy stromu jsou ohodnoceny logickými proměnnými.

1.1.4 Podformule. Ze syntaktického stromu formule α jednoduše poznáme všechny její podformule: *Podformule* formule α jsou všechny formule odpovídající podstromům syntaktického stromu formule α.

1.2 Pravdivostní ohodnocení

1.2.1 *Pravdivostní ohodnocení*, též pouze *ohodnocení formulí*, je zobrazení $u\colon \mathcal{P}(A) \to \{0,1\}$, které splňuje pravidla

(1) $\neg\alpha$ je pravdivá právě tehdy, když α je nepravdivá, tj $u(\neg\alpha) = 1$ právě tehdy, když $u(\alpha) = 0$;

(2) $\alpha\wedge\beta$ je pravdivá právě tehdy, když α a β jsou obě pravdivé, tj. $u(\alpha\wedge\beta) = 1$ právě tehdy, když $u(\alpha) = u(\beta) = 1$;

(3) $\alpha \vee \beta$ je nepravdivá právě tehdy, když α a β jsou obě nepravdivé, tj. $u(\alpha \vee \beta) = 0$ právě tehdy, když $u(\alpha) = u(\beta) = 0$;

(4) $\alpha \Rightarrow \beta$ je nepravdivá právě tehdy, když α je pravdivá a β nepravdivá, tj. $u(\alpha \Rightarrow \beta) = 0$ právě tehdy, když $u(\alpha) = 1$ a $u(\beta) = 0$;

(5) $\alpha \Leftrightarrow \beta$ je pravdivá právě tehdy, když buď obě formule α a β jsou pravdivé nebo obě jsou nepravdivé tj. $u(\alpha \Leftrightarrow \beta) = 1$ právě tehdy, když $u(\alpha) = u(\beta)$.

1.2.2 Pravdivostní tabulky. Vlastnosti, které ohodnocení formulí musí mít, znázorňujeme též pomocí tzv. pravdivostních tabulek logických spojek. Jsou to:

α	$\neg\alpha$
0	1
1	0

α	β	$\alpha \wedge \beta$	$\alpha \vee \beta$	$\alpha \Rightarrow \beta$	$\alpha \Leftrightarrow \beta$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

1.2.3 Věta. Každé zobrazení $u_0\colon A \to \{0,1\}$ jednoznačně určuje ohodnocení $u : \mathcal{P}(A) \to \{0,1\}$ takové, že $u_0(a) = u(a)$ pro všechna $a \in A$.

1.2.4 Důsledek. Dvě ohodnocení $u, v\colon \mathcal{P}(A) \to \{0,1\}$ jsou shodná právě tehdy, když pro všechny logické proměnné $x \in A$ platí $u(x) = v(x)$.

1.2.5 Tautologie, kontradikce, splnitelné formule. Formule se nazývá *tautologie*, jestliže je pravdivá ve všech ohodnoceních formulí; nazývá se *kontradikce*, jestliže je nepravdivá ve všech ohodnoceních formulí. Formule je *splnitelná*, jestliže existuje aspoň jedno ohodnocení formulí, ve kterém je pravdivá.

1.2.6 Příklady

1. Formule $\alpha \vee \neg\alpha$, $\alpha \Rightarrow \alpha$, $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ jsou tautologie.
2. Formule $a \vee b$, $(a \Rightarrow b) \Rightarrow a$ jsou splnitelné, ale ne tautologie.
3. Formule $\alpha \wedge \neg\alpha$ je kontradikce. Kontradikce je také každá negace tautologie.

1.3 Tautologická ekvivalence

1.3.1 Tautologická ekvivalence formulí. Řekneme, že formule φ a ψ jsou *tautologicky ekvivalentní* (také *sémanticky ekvivalentní*), jestliže pro každé ohodnocení u platí $u(\varphi) = u(\psi)$.

1.3.2 Tvrzeni. Pro každé formule α, β a γ platí:

- $\alpha \models\!\models \alpha$,
- je-li $\alpha \models\!\models \beta$, pak i $\beta \models\!\models \alpha$,
- je-li $\alpha \models\!\models \beta$ a $\beta \models\!\models \gamma$, pak i $\alpha \models\!\models \gamma$.

Jsou-li α, β, γ a δ formule splňující $\alpha \models\!\models \beta$ a $\gamma \models\!\models \delta$, pak platí

- $\neg\alpha \models\!\models \neg\beta$;
- $(\alpha \wedge \gamma) \models\!\models (\beta \wedge \delta)$, $(\alpha \vee \gamma) \models\!\models (\beta \vee \delta)$, $(\alpha \Rightarrow \gamma) \models\!\models (\beta \Rightarrow \delta)$, $(\alpha \Leftrightarrow \gamma) \models\!\models (\beta \Leftrightarrow \delta)$.

1.3.3 Poznámka. Z vlastností z předchozího tvrzení např. platí: Protože $a \Rightarrow b \models\!\models (\neg a \vee b)$, je také

$$\varphi = ((c \wedge d) \Rightarrow \neg(a \Rightarrow b)) \wedge (c \vee \neg b) \models\!\models ((c \wedge d) \Rightarrow \neg(\neg a \vee b)) \wedge (c \vee \neg b) = \psi.$$

Zhruba řečeno jsme „dosadili“ do φ místo $a \Rightarrow b$ formuli $\neg a \vee b$ a dostali jsme formuli ψ tautologicky ekvivalentní s φ. Takovéto „dosazování“ předchozí tvrzení umožňuje.

1.3.4 Příklad. Pro každou formuli α je formule $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ tautologie.
Ano, máme

$$\alpha \Rightarrow (\beta \Rightarrow \alpha) \models\!\models \neg\alpha \vee (\neg\beta \vee \alpha) \models\!\models (\neg\alpha \vee \alpha) \vee \neg\beta,$$

kde poslední formule je tautologie.

1.3.5 Tvrzení. Pro každé formule α, β a γ platí

- $\alpha \wedge \alpha \models\!\models \alpha$, $\alpha \vee \alpha \models\!\models \alpha$ (idempotence $\wedge$ a $\vee$);
- $\alpha \wedge \beta \models\!\models \beta \wedge \alpha$, $\alpha \vee \beta \models\!\models \beta \vee \alpha$ (komutativita $\wedge$ a $\vee$);
- $\alpha \wedge (\beta \wedge \gamma) \models\!\models (\alpha \wedge \beta) \wedge \gamma$, $\alpha \vee (\beta \vee \gamma) \models\!\models (\alpha \vee \beta) \vee \gamma$ (asociativita $\wedge$ a $\vee$);
- $\alpha \wedge (\beta \vee \alpha) \models\!\models \alpha$, $\alpha \vee (\beta \wedge \alpha) \models\!\models \alpha$ (absorpce $\wedge$ a $\vee$);
- $\neg\neg\alpha \models\!\models \alpha$;
- $\neg(\alpha \wedge \beta) \models\!\models (\neg\alpha \vee \neg\beta)$, $\neg(\alpha \vee \beta) \models\!\models (\neg\alpha \wedge \neg\beta)$ (de Morganova pravidla);
- $\alpha \wedge (\beta \vee \gamma) \models\!\models (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$, $\alpha \vee (\beta \wedge \gamma) \models\!\models (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ (distributivní zákony).
- $(\alpha \Rightarrow \beta) \models\!\models (\neg\alpha \vee \beta)$.

Je-li navíc $\mathbf{T}$ libovolná tautologie a $\mathbf{F}$ libovolná kontradikce, pak

- $\mathbf{T} \wedge \alpha \models\!\models \alpha$, $\mathbf{T} \vee \alpha \models\!\models \mathbf{T}$, $\mathbf{F} \wedge \alpha \models\!\models \mathbf{F}$, $\mathbf{F} \vee \alpha \models\!\models \alpha$;
- $\alpha \wedge \neg\alpha \models\!\models \mathbf{F}$, $\alpha \vee \neg\alpha \models\!\models \mathbf{T}$.

1.3.6 Poznámka. Platí $\alpha \models\!\models \beta$ právě tehdy, když $\alpha \Leftrightarrow \beta$ je tautologie.

1.3.7 Další spojky. Každá formule s jednou (nebo žádnou) logickou proměnnou představuje zobrazení z množiny $\{0,1\}$ do množiny $\{0,1\}$. Existují čtyři taková zobrazení:

x	f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1

Funkce f_1 je konstantní 0, jedná se o kontradikci a budeme ji značit **F**. Podobě funkce f_4 je tautologie (konstatní 1), značíme je **T**. Funkce f_2 je vlastně logická proměnná x a funkce f_3 je $\neg x$. Tedy nemáme „další“ unární spojky.

1.3.8 Další binární spojky. Každá formule s nejvýše dvěma logickými proměnnými představuje zobrazení z množiny $\{0,1\}^2$ do množiny $\{0,1\}$. Existuje šestnáct takových zobrazení:

x	y	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Každá z funkcí $f_0, \ldots, f_{15}$ odpovídá nějaké formuli; např. f_0 odpovídá kontradikci **F**, f_{15} tautologii **T**, f_1 formuli $x \wedge y$, funkce f_{10} formuli $\neg y$, atd. Jako „nové“ spojky se zavádějí následující zkratky za formule odpovídající funkcím f_6, f_8 a f_{14}.

1.3.9 NAND. Logická spojka $|$, nazývaná *NAND* (také *Shefferův operátor*), je definována

$$x \mid y \models\!\models \neg(x \wedge y).$$

Odpovídá funkci f_{14}.

1.3.10 NOR. Logická spojka $\downarrow$, nazývaná *NOR* (také *Peirceova šipka*), je definována

$$x \downarrow y \models\!\models \neg(x \vee y).$$

Odpovídá funkci f_8.

1.3.11 XOR. Logická spojka $\oplus$, nazývaná *XOR* (také *vylučovací nebo*), je definována

$$x \oplus y \models\!\models \neg(x \Leftrightarrow y).$$

Odpovídá funkci f_6.

1.3.12 Tvrzení. Pro každou formuli α existují formule $\beta_1, \beta_2, \beta_3, \beta_4$ takové, že všechny čtyři jsou tautologicky ekvivalentní s formulí α a

- β_1 obsahuje jen spojky $\neg$ a $\Rightarrow$;
- β_1 obsahuje jen spojky $\neg$ a $\wedge$;
- β_1 obsahuje jen spojky $\neg$ a $\vee$.
- β_4 obsahuje pouze spojku $|$.

1.4 CNF a DNF

Každé formuli o n logických proměnných odpovídá pravdivostní tabulka. Na tuto tabulku se můžeme dívat jako na zobrazení, které každé n-tici 0 a 1 přiřazuje 0 nebo 1. Ano, řádek pravdivostní tabulky je popsán n-ticí 0 a 1, hodnota je pak pravdivostní hodnota formule pro toto dosazení za logické proměnné. Zobrazení z množiny všech n-tic 0 a 1 do množiny $\{0,1\}$ se nazývá *Booleova funkce*. Naopak platí, že pro každou Booleovu funkci existuje formule, která této funkci odpovídá. Ukážeme v dalším, že dokonce můžeme volit formu ve speciálním tvaru, v tzv. *konjunktivním normálním tvaru* a *disjunktivním normálním tvaru*.

1.4.1 Booleova funkce. *Booleovou funkcí n proměnných*, kde n je přirozené číslo, rozumíme každé zobrazení $f\colon\{0,1\}^n \to \{0,1\}$, tj. zobrazení, které každé n-tici $(x_1, x_2, \ldots, x_n)$ nul a jedniček přiřazuje nulu nebo jedničku (označenou $f(x_1, x_2, \ldots, x_n)$).

1.4.2 Disjunktivní normální tvar. *Literál* je logická proměnná nebo negace logické proměnné. Řekneme, že formule je v *disjunktivním normálním tvaru*, zkráceně v *DNF*, jestliže je disjunkcí jedné nebo několika formulí, z nichž každá je literálem nebo konjunkcí literálů.

Poznamenejme, že literálu nebo konjunkci literálů se také říká *minterm*. Jestliže každý minterm obsahuje všechny proměnné, říkáme, že se jedná o *úplnou DNF*.

1.4.3 Věta. Ke každé Booleově funkci f existuje formule v DNF odpovídající f.

1.4.4 Zdůvodnění. Jestliže funkce f má všechny hodnoty rovny nule, je jí odpovídající formule kontradikce. Kontradikci můžeme reprezentovat např. formulí $x \wedge \neg x$.

Předpokládejme, že funkce f není konstantní 0, tj. pro aspoň jednu n-tici $x_1\, x_2\, \ldots\, x_n$ má hodnotu 1. Pro každou takovou n-tici utvoříme minterm, který tuto n-tici popisuje takto: je-li $x_i = 1$ dáme do mintermu literál x_i, je-li $x_i = 0$ dáme do mintermu literál $\neg x_i$. (Např. pro n-tici, kde $x_1 = 0$ a ostatní $x_i = 1$ má odpovídající minterm α tvar $\alpha = \neg x_1 \wedge x_2 \wedge \ldots \wedge x_n$. Uvědomte si, že α je formule, která je pravdivá pouze v jednom ohodnocení; a to $u(x_1) = 0$, $u(x_2) = u(x_3) = \ldots = u(x_n) = 1$.) Výsledná formule je pak disjunkcí všech mintermů odpovídajících n-ticím, ve kterých je hodnota funkce f rovna 1.

Poznamenejme, že takto vzniklá formule je úplná DNF.

1.4.5 Důsledek. Ke každé formuli α existuje formule β, která je v DNF a navíc $\alpha \models\!\!\models \beta$.

1.4.6 Konjunktivní normální tvar. Řekneme, že formule je v *konjunktivním normálním tvaru*, zkráceně v *CNF*, jestliže je konjunkcí jedné nebo několika formulí, z nichž každá je literálem nebo disjunkcí literálů.

Poznamenejme, že literálu nebo disjunkci literálů se také říká *maxterm* nebo *klausule*. Jestliže každá klausule obsahuje všechny proměnné, říkáme, že se jedná o *úplnou CNF*.

1.4.7 Věta. Ke každé Booleově funkci f existuje formule v CNF odpovídající f.

1.4.8 Zdůvodnění. Jestliže funkce f má všechny hodnoty rovny 1, je jí odpovídající formule tautologie. Tautologii můžeme reprezentovat např. formulí $x \vee \neg x$.

Předpokládejme, že funkce f není konstantní 1, tj. pro aspoň jednu n-tici $x_1\, x_2\, \ldots\, x_n$ má hodnotu 0. Vytvoříme funkci f' takovou, že má přesně opačné hodnoty než f. Tj. $f'(x_1\, x_2\, \ldots\, x_n) = 1$ právě tehdy, když $f(x_1\, x_2\, \ldots\, x_n) = 0$. K funkci f' najdeme formuli β v DNF podle 1.4.3. Hledanou formuli α v CNF odpovídající funkci f dostaneme z formule $\neg\beta$ dvojím použitím de Morganova zákona.

Poznamenejme, že takto vzniklá formule je úplná CNF.

1.4.9 Důsledek. Ke každé formuli α existuje formule β, která je v CNF a navíc $\alpha \models\!\!\!\models \beta$.

1.4.10 Karnaughovy mapy. Pro zjednodušení formulí v disjunktivní a konjunktivní normální formě (viz 1.4.3 a 1.4.7) je možné použít tzv. Karnaughovy mapy. Vhodné je to hlavně pro dvě nebo tři logické proměnné.

1.5 Booleovský kalkul.

Víme, že pro pravdivostní ohodnocení formulí platí:

$$\begin{aligned} u(a \vee b) &= \max\{u(a), u(b)\} = \max\{x, y\}, \\ u(a \wedge b) &= \min\{u(a), u(b)\} = \min\{x, y\}, \\ u(\neg a) &= 1 - u(a) = 1 - x. \end{aligned}$$

kde $x = u(a)$, $y = u(b)$.

1.5.1 Booleovské operace. To motivuje zavedení booleovských operací (pro hodnoty 0, 1):

$$\begin{array}{lrcl} \text{součin} & x \cdot y & = & \min\{x, y\}, \\ \text{logický součet} & x + y & = & \max\{x, y\}, \\ \text{doplněk} & \overline{x} & = & 1 - x. \end{array}$$

Pro tyto operace platí řada rovností, tak, jak je známe z výrokové logiky:

1.5.2 Tvrzení. Pro všechna $x, y, z \in \{0, 1\}$ platí:

1. $x \cdot x = x,\ x + x = x$;
2. $x \cdot y = y \cdot x,\ x + y = y + x$;
3. $x \cdot (y \cdot z) = (x \cdot y) \cdot z,\ x + (y + z) = (x + y) + z$;
4. $x \cdot (y + x) = x,\ x + (y \cdot x) = x$;
5. $x \cdot (y + z) = (x \cdot y) + (x \cdot z),\ x + (y \cdot z) = (x + y) \cdot (x + z)$;

6. $\overline{\overline{x}} = x$;
7. $\overline{x+y} = \overline{x} \cdot \overline{y},\ \overline{x \cdot y} = \overline{x} + \overline{y}$;
8. $x + \overline{x} = 1,\ x \cdot \overline{x} = 0$;
9. $x \cdot 0 = 0,\ x \cdot 1 = x$;
10. $x + 1 = 1,\ x + 0 = x$.

1.5.3 Booleovy funkce v DNF a CNF. Nyní můžeme pro Booleovu funkci psát pomocí výše uvedených operaci, např.

$$f(x,y,z) = \overline{x}\,\overline{y}\,\overline{z} + \overline{x}\,\overline{y}\,z\ + \overline{x}\,y\,\overline{z} + \overline{x}\,y\,z\ + x\,\overline{y}\,z$$

a říkat, že jsme Booleovu funkci napsali v *disjunktivní normální formě*. Rovnost opravdu platí; dosadíme-li za logické proměnné jakékoli hodnoty, pak pravá strana rovnosti určuje hodnotu Booleovy funkce f. Obdobně jako jsme Booleovu funkci f napsali v disjunktivní normální formě, můžeme ji také napsat v *konjunktivní normální formě* a to takto:

$$f(x,y,z) = (\overline{x} + y + z)\,(\overline{x} + \overline{y} + z)\,(\overline{x} + \overline{y} + \overline{z}).$$

1.5.4 Věta. Každou Booleovu funkci lze napsat v disjunktivní normální formě i v konjunktivní normální formě.

1.6 Sémantický důsledek

1.6.1 Množina formulí pravdivá v ohodnocení. Řekneme, že množina formulí S je *pravdivá* v ohodnocení u, jestliže každá formule z S je pravdivá v u, tj. je-li $u(\varphi) = 1$ pro všechna $\varphi \in S$. Množina formulí S je nepravdivá v ohodnocení u, jestliže existuje formule $\varphi \in S$, která je nepravdivá v ohodnocení u.

Fakt, že množina formulí S je pravdivá v ohodnocení u zapisujeme též $u(S) = 1$, fakt, že S je nepravdivá v u, zapisujeme také $u(S) = 0$.

1.6.2 Poznámka. Prázdná množina formulí je pravdivá v každém ohodnocení.

Ano, každá množina je v daném ohodnocení u buď pravdivá nebo nepravdivá. Protože prázdná množina nemůže být v ohodnocení u nepravdivá, musí v něm být pravdivá.

1.6.3 Splnitelná množina formulí. Řekneme, že množina formulí S je *splnitelná*, jestliže existuje pravdivostní ohodnocení u, v němž je S pravdivá. V opačném případě se množina S nazývá *nesplnitelná*.

Poznamenejme, že prázdná množina formulí je splnitelná.

1.6.4 Sémantický důsledek. Řekneme, že formule φ je *konsekventem*, též *sémantickým* nebo *tautologickým důsledkem* množiny formulí S, jestliže φ je pravdivá v každém ohodnocení u, v němž je pravdivá S.

Fakt, že formule φ je konsekventem množiny S, označujeme $S \models \varphi$. Je-li množina S prázdná, píšeme $\models \varphi$ místo $\emptyset \models \varphi$. Je-li množina S jednoprvková, tj. $S = \{\alpha\}$, píšeme $\alpha \models \varphi$ místo $\{\alpha\} \models \varphi$.

1.6.5 Poznámka. Definici sémantického důsledku můžeme přeformulovat následovně:

$$S \models \varphi \quad \text{iff} \quad u(S) \leq u(\varphi) \text{ pro každé ohodnocení } u.$$

1.6.6 Příklady. Pro každé formule α, β, γ platí

1. $\{\alpha, \alpha \Rightarrow \beta\} \models \beta$.
2. $\{\alpha \Rightarrow \beta, \neg\beta\} \models \neg\alpha$.
3. $\{\alpha \vee \beta, \alpha \Rightarrow \gamma, \beta \Rightarrow \gamma\} \models \gamma$.
4. $\{\alpha \Rightarrow \beta, \alpha \Rightarrow \neg\beta\} \models \neg\alpha$.
5. $\{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma\} \models (\alpha \Rightarrow \gamma)$.
6. $\{(\alpha \wedge \beta) \Rightarrow \gamma, (\alpha \wedge \neg\beta) \Rightarrow \gamma\} \models (\alpha \Rightarrow \gamma)$.

1.6.7 Tvrzení.

1. Je-li S množina formulí a $\varphi \in S$, pak φ je konsekventem S, tj. $S \models \varphi$ pro každou $\varphi \in S$.
2. Tautologie je konsekventem každé množiny formulí S.
3. Formule φ je tautologie právě tehdy, když $\models \varphi$.
4. Každá formule je konsekventem nesplnitelné množiny formulí.
5. Máme dvě množiny formulí M a N, kde $M \subseteq N$. Pak každý konsekvent množiny M je také konsekventem množiny N, tj. je-li $M \models \varphi$, pak $N \models \varphi$.
6. Je-li φ konsekventem množiny formulí $\{\alpha_1, \ldots, \alpha_k\}$ a každá formule α_i je konsekventem množiny formulí S, pak φ je konsekventem S.

1.6.8 Poznámka. Uvědomme si, že $\alpha \models\!\!\!\models \beta$ právě tehdy, když platí současně $\alpha \models \beta$ a také $\beta \models \alpha$.

1.6.9 Tvrzení. Pro každé dvě formule α a β platí:

$$\alpha \models \beta \quad \text{právě tehdy, když} \quad \alpha \Rightarrow \beta \text{ je tautologie.}$$

1.6.10 Věta. Pro množinu formulí S a formuli φ platí

$$S \models \varphi \quad \text{právě tehdy, když} \quad S \cup \{\neg\varphi\} \quad \text{je nesplnitelná.}$$

1.6.11 Věta o dedukci. Pro množinu formulí S a formule φ a ψ platí

$$S \cup \{\varphi\} \models \psi \quad \text{právě tehdy, když} \quad S \models (\varphi \Rightarrow \psi).$$

1.7 Rezoluční metoda ve výrokové logice

Rezoluční metoda rozhoduje, zda daná množina klausulí je splnitelná nebo je nesplnitelná. Tím je také "universální metodou"pro řešení základních problémů ve výrokové logice, neboť:

1. Daná formule φ je sémantickým důsledkem množiny formulí S právě tehdy, když množina $S \cup \{\neg\varphi\}$ je nesplnitelná.
2. Ke každé formuli α existuje množina klausulí S_α taková, že α je pravdivá v ohodnocení u právě tehdy, když v tomto ohodnocení je pravdivá množina S_α.

1.7.1 Klausule. Množinu všech logických proměnných označíme A. Připoměňme, že *literál* je buď logická proměnná (tzv. *positivní literál*) nebo negace logické proměnné (tzv. *negativní literál*). *Komplementární literály* jsou literály p a $\neg p$. *Klausule* je literál nebo disjunkce konečně mnoha literálů (tedy i žádného).

Zvláštní místo mezi klausulemi zaujímá *prázdná klausule*, tj. klausule, která neobsahuje žádný literál a tudíž se jedná o kontradikci. Proto ji budeme označovat F.

Pro jednoduchost zavedeme následující konvenci: Máme dánu klausuli C a literál p, který se v C vyskytuje. Pak symbolem $C \setminus p$ označujeme klausuli, která obsahuje všechny literály jako C kromě p. Tedy např. je-li $C = \neg x \vee y \vee \neg z$, pak

$$C \setminus \neg z = \neg x \vee y.$$

1.7.2 Rezolventa. Řekneme, že klausule D je *rezolventou klausulí* C_1 a C_2 právě tehdy, když existuje literál p takový, že p se vyskytuje v klausuli C_1, $\neg p$ se vyskytuje v klausuli C_2 a

$$D = (C_1 \setminus p) \vee (C_2 \setminus \neg p).$$

Také říkáme, že klausule D je *rezolventou* C_1 *a* C_2 *podle literálu* p a značíme $D = res_p(C_1, C_2)$.

1.7.3 Tvrzení. Máme dány dvě klausule C_1, C_2 a označme D jejich rezolventu. Pak D je sémantický důsledek množiny $\{C_1, C_2\}$.

1.7.4 Tvrzení. Máme dánu množinu klausulí S a označme D rezolventu některých dvou klausulí z množiny S. Pak množiny S a $S \cup \{D\}$ jsou pravdivé ve stejných ohodnoceních.

1.7.5 Rezoluční princip. Označme

$$\begin{aligned} R(S) &= S \cup \{D \mid D \text{ je rezolventa některých klausulí z } S\} \\ R^0(S) &= S \\ R^{i+1}(S) &= R(R^i(S)) \quad \text{pro} \quad i \in \mathbb{N} \\ R^\star(S) &= \bigcup\{R^i(S) \mid i \geq 0\}. \end{aligned}$$

Protože pro konečnou množinu logických proměnných existuje jen konečně mnoho klausulí, musí existovat n takové, že $R^n(S) = R^{n+1}(S)$. Pro toto n platí $R^n(S) = R^\star(S)$.

1.7.6 Věta (Rezoluční princip.) Množina klausulí S je splnitelná právě tehdy, když $R^\star(S)$ neobsahuje prázdnou klausuli F.

1.7.7 Základní postup. Předchozí věta dává návod, jak zjistit, zda daná množina klausulí je spnitelná nebo je nesplnitelná:

1. Formule množiny M převedeme do CNF a M pak nahradíme množinou S všech klausulí vyskytujících se v některé formuli v CNF. Klausule, které jsou tautologiemi, vynecháme. Jestliže nám nezbyde žádná klausule, množina M se skládala z tautologií a je pravdivá v každém pravdivostním ohodnocení.
2. Vytvoříme $R^\star(S)$.
3. Obsahuje-li $R^\star(S)$ prázdnou klausuli, je množina S (a tedy i množina M) nesplnitelná, v opačném případě je M splnitelná.

Je zřejmé, že konstrukce celé množiny $R^\star(S)$ může být zbytečná — stačí pouze zjistit, zda $R^\star(S)$ obsahuje F.

1.7.8 Výhodnější postup. Existuje ještě jeden postup, který usnadní práci s použitím rezoluční metody. Ten nejenom že nám odpoví na otázku, zda konečná množina klausulí S je splnitelná nebo nesplnitelná, ale dokonce nám umožní v případě splnitelnosti sestrojit aspoň jedno pravdivostní ohodnocení, v němž je množina S pravdivá.

Máme konečnou množinu klausulí S, kde žádná klausule není tautologií. Zvolíme jednu logickou proměnnou (označme ji x), která se v některé z klausulí z S vyskytuje. Najdeme množinu klausulí S_1 s těmito vlastnostmi:

1. Žádná klausule v S_1 neobsahuje logickou proměnnou x.
2. Množina S_1 je splnitelná právě tehdy, když je splnitelná původní množina S.

Množinu S_1 vytvoříme takto: Rozdělíme klausule množiny S do tří skupin:

M_0 se skládá ze všech klausulí množiny S, které neobsahují logickou proměnnou x.

M_x se skládá ze všech klausulí množiny S, které obsahují positivní literál x.

$M_{\neg x}$ se skládá ze všech klausulí množiny S, které obsahují negativní literál $\neg x$.

Označme N množinu všech rezolvent klausulí množiny S podle literálu x; tj. rezolvent vždy jedné klausule z množiny M_x s jednou klausulí z množiny $M_{\neg x}$. Všechny tautologie vyřadíme.

Položíme $S_1 = M_0 \cup N$.

1.7.9 Tvrzení. Množina klausulí S_1 zkonstruovaná výše je splnitelná právě tehdy, když je splnitelná množina S.

1.7.10 Dostali jsme tedy množinu klausulí S_1, která již neobsahuje logickou proměnnou x a je splnitelná právě tehdy, když je splnitelná množina S. Navíc, množina S_1 má o jednu logickou proměnnou méně než množina S.

Nyní opakujeme postup pro množinu S_1. Postup skončí jedním ze dvou možných způsobů:

1. Při vytváření rezolvent dostaneme prázdnou kalusuli F. Tedy S je nesplnitelná.

2. Dostaneme prázdnou množinu klausulí. V tomto případě je množina S splnitelná.

1.7.11 Je výhodné předchozí postup znázorňovat v tabulce. Na začátku práce utvoříme tabulku, která obsahuje pro každou klausuli množiny S jeden sloupec. V prvním řádku vybereme jednu proměnnou, řekněme x, a řádek označíme x. Procházíme neoznačené sloupce tabulky, které odpovídají klausulím obsahujícím proměnnou x. Ve sloupci do řádku napíšeme 1, v případě, že klasusule obsahuje literál x, nebo 0, v případě, že klausule obsahuje literál $\neg x$.

Vybereme libovolnou klausuli C_1, která má v řádku 1, a libovolnou klausuli C_2, která má v řádku 0. Sloupec pro jejich rezolventu podle x přidáme v případě, že se jedná o novou klausuli, která není tautologií. Jestliže žádný sloupece není v řádku označen 1 ($M_x = \emptyset$) nebo žádný sloupec není v řádku označen 0 ($M_{\neg x} = \emptyset$), nepřidáváme nic.

Jestliže jsme přidali prázdnou klausuli, výpočet končí, množina S je nesplnitelná. Jestliže každý sloupec již má 1 nebo 0, výpočet ukončíme, množina S je splnitelná. Tím jsme ukončili první krok.

Ve druhém kroku se zajímáme jen o sloupce tabulky, které nemají ještě ani číslo 1 ani 0 (tyto sloupce tvoří množiny S_1). Opět vybereme proměnnou, která se v některé ze zbylých klausulí vyskytuje. Postupujeme dále jako v kroku 1.

Celý postup tedy končí buď přidáním prázdné klausule, v tom případě je množina S nesplnitelná, nebo vyčerpáním neoznačených sloupců, v tomto případě je množina S splnitelná.

Je-li množina S splnitelná, tak jedno pravdivostní ohodnocení, ve kterém je množina S pravdivá, dostaneme zpětným postupem v tabulce.

Kapitola 2

Predikátová logika

2.1 Syntaxe predikátové logiky

Nejprve zavedeme syntaxi predikátové logiky, tj. uvedeme pravidla, podle nichž se tvoří syntakticky správné formule predikátové logiky. Význam a pravdivostní hodnota nás bude zajímat až dále.

Správně utvořené formule budou řetězce (posloupnosti) symbolů tzv. *jazyka predikátové logiky.*

2.1.1 Jazyk predikátové logiky $\mathcal{L}$. *Jazyk predikátové logiky* se skládá z

1. *logických symbolů*, tj.:
   a) spočetné množiny individuálních proměnných: $\mathsf{Var} = \{x, y, \ldots, x_1, x_2, \ldots\}$
   b) výrokových logických spojek: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
   c) obecného kvantifikátoru $\forall$ a existenčního kvantifikátoru $\exists$
2. *speciálních symbolů*, tj.:
   a) množiny Pred predikátových symbolů (nesmí být prázdná)
   b) množiny Kons konstantních symbolů (může být prázdná)
   c) množiny Func funkčních symbolů (může být prázdná)
3. pomocných symbolů, jako jsou závorky „(, [,) ,]“ a čárka „,“.

Pro každý predikátový i funkční symbol máme dáno přirozené číslo n kolika objektů se daný predikát týká, nebo kolika proměnných je daný funkční symbol. Tomuto číslu říkáme *arita* nebo též *četnost* predikátového symbolu nebo funkčního symbolu. Funkční symboly mají aritu větší nebo rovnu 1, predikátové symboly připouštíme i arity 0.

2.1.2 Poznámka. Predikátové symboly budeme většinou značit velkými písmeny, tj. např. $P, Q, R, \ldots, P_1, P_2, \ldots$; konstantní symboly malými písmeny ze začátku abecedy, tj. $a, b, c, \ldots, a_1, \ldots$, a funkční symboly většinou $f, g, h, \ldots, f_1, f_2, \ldots$. Formule predikátové logiky budeme označovat malými řeckými písmeny (obdobně, jako jsme to dělali pro výrokové formule). Kdykoli se od těchto konvencí odchýlíme, tak v textu na to upozorníme.

Poznamejme, že přestože často budeme mluvit o n-árních predikátových symbolech a n-árních funkčních symbolech, v běžné praxi se setkáme jak s predikáty, tak funkcemi arity nejvýše tři. Nejběžnější jsou predikáty a funkční symboly arity 1, těm říkáme též *unární*, nebo arity 2, těm říkáme též *binární*.

Predikátové symboly arity 0 představují nestrukturované výroky (netýkají se žádného objektu). Tímto způsobem se v predikátové logice dá popsat i výrok: „Prší“.

Poznamenejme ještě, že někteří autoři konstantní symboly zahrnují pod nulární funkční symboly (tj. funkční symboly arity 0).

2.1.3 Termy. Množina *termů* je definována těmito pravidly:

1. Každá proměnná a každý konstantní symbol je term.
2. Jestliže f je funkční symbol arity n a $t_1, t_2, \ldots, t_n$ jsou termy, pak $f(t_1, t_2, \ldots, t_n)$ je také term.
3. Nic, co nevzniklo konečným použitím pravidel 1 a 2, není term.

2.1.4 Poznámka. Term je zhruba řečeno objekt, pouze může být složitěji popsán než jen proměnnou nebo konstantou. V jazyce predikátové logiky termy vystupují jako „podstatná jména“.

2.1.5 Atomické formule. *Atomická formule* je predikátový symbol P aplikovaný na tolik termů, kolik je jeho arita. Jinými slovy, pro každý predikátový symbol $P \in \mathsf{Pred}$ arity n a pro každou n-tici termů $t_1, t_2, \ldots, t_n$ je $P(t_1, t_2, \ldots, t_n)$ atomická formule.

2.1.6 Formule. Množina *formulí* je definována těmito pravidly:

1. Každá atomická formule je formule.
2. Jsou-li φ a ψ dvě formule, pak $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$, $(\varphi \Leftrightarrow \psi)$ jsou opět formule.
3. Je-li φ formule a x proměnná, pak $(\forall x\, \varphi)$ a $(\exists x\, \varphi)$ jsou opět formule.
4. Nic, co nevzniklo pomocí konečně mnoha použití bodů 1 až 3, není formule.

2.1.7 Poznámka. Formule predikátové logiky jsme definovali obdobně jako výrokové formule: Nejprve jsme definovali „ty nejjednodušší“ formule (atomické formule) a potom pomocí logických spojek a kvatifikátorů konstruujeme složitější formule. Ve výrokové logice byl první krok daleko jednodušší, protože atomické výroky (logické proměnné) nebyly strukturované. Vlastní konstrukce formulí je však v obou případech podobná.

2.1.8 Konvence.

1. Úplně vnější závorky nepíšeme. Píšeme tedy např. $(\exists x P(x)) \vee R(a, b)$ místo $((\exists x\, P(x)) \vee R(a, b))$.
2. Spojka „negace“ má vždy přednost před výrokovými logickými spojkami a proto píšeme např. $\forall x\, (\neg P(x) \Rightarrow Q(x))$ místo $\forall x\, ((\neg P(x)) \Rightarrow Q(x))$.

2.1.9 Syntaktický strom formule. Ke každé formuli predikátové logiky můžeme přiřadit její *syntaktický strom* (též *derivační strom*) podobným způsobem jako jsme to udělali v případě výrokových formulí. Rozdíl je v tom, že kvantifikátory považujeme za unární (tj. mají pouze jednoho následníka) a také pro termy vytváříme jejich syntaktický strom. Listy syntaktického stromu jsou vždy ohodnoceny buď proměnnou nebo konstantou.

2.1.10 Podformule. *Podformule* formule φ je libovolný podřetězec φ, který je sám formulí. Jinými slovy: Podformule formule φ je každý řetězec odpovídající podstromu syntaktického stromu formule φ, určenému vrcholem ohodnoceným predikátovým symbolem, logickou spojkou nebo kvantifikátorem.

2.1.11 Volný a vázaný výskyt proměnné. Máme formuli φ a její syntaktický strom. List syntaktického stromu obsazený proměnnou x je výskyt proměnné x ve formuli φ. Výskyt proměnné x je *vázaný* ve formuli φ, jestliže při postupu od listu ohodnoceného tímto x ve směru ke kořeni syntaktického stromu narazíme na kvantifikátor s touto proměnnou. V opačném případě mluvíme o *volném* výskytu proměnné x.

2.1.12 Sentence. Formule, která má pouze vázané výskyty proměnné, se nazývá *sentence*, též *uzavřená formule*. Formuli, která má pouze volné výskyty proměnné, se říká *otevřená formule*.

2.1.13 Legální přejmenování proměnné. Přejmenování výskytů proměnné x ve formuli φ je *legálním* přejmenováním proměnné, jestliže

- jedná se o výskyt vázané proměnné ve φ;
- přejmenováváme všechny výskyty x vázané daným kvantifikátorem;
- po přejmenování se žádný dříve volný výskyt proměnné nesmí stát vázaným výskytem.

2.1.14 Rovnost formulí. Dvě formule považujeme za *stejné*, jestliže se liší pouze legálním přejmenováním vázaných proměnných.

Každou formuli φ lze napsat tak, že každá proměnná má ve formuli buď jen volné výskyty nebo jen vázané výskyty.

2.2 Sémantika predikátové logiky

Nyní se budeme zabývat sémantikou formulí, tj. jejich významem a pravdivostí.

2.2.1 Interpretace jazyka predikátové logiky. *Interpretace* predikátové logiky s predikátovými symboly Pred, konstantními symboly Kons a funkčními symboly Func je dvojice $\langle U, [\![-]\!] \rangle$, kde

- U je neprázdná množina nazývaná *universum*;
- $[\![-]\!]$ je přiřazení, které
  1. každému predikátovému symbolu $P \in \mathsf{Pred}$ arity n přiřazuje podmnožinu $[\![P]\!]$ množiny U^n, tj. n-ární relaci na množině U.

2. každému konstantnímu symbolu $a \in \mathsf{Kons}$ přiřazuje prvek z U, značíme jej $[\![a]\!]$,
3. každému funkčnímu symbolu $f \in \mathsf{Func}$ arity n přiřazuje zobrazení množiny U^n do U, značíme je $[\![f]\!]$,

Množina U se někdy nazývá *domain* a označuje D.

2.2.2 Kontext proměnných. Je dána interpretace $\langle U, [\![-]\!]\rangle$. *Kontext proměnných* je zobrazení ρ, které každé proměnné $x \in \mathsf{Var}$ přiřadí prvek $\rho(x) \in U$. Je-li ρ kontext proměnných, $x \in \mathsf{Var}$ a $d \in U$, pak

$$\rho[x := d]$$

označuje kontext proměnných, který má stejné hodnoty jako ρ, a liší se pouze v proměnné x, kde má hodnotu d. Kontextu proměnných $\rho[x := d]$ též říkáme *update* kontextu ρ o hodnotu d v x.

2.2.3 Interpretace termů při daném kontextu proměnných. Je dána interpretace $\langle U, [\![-]\!]\rangle$ a kontext proměnných ρ. Pak termy interpretujeme následujícím způsobem.

1. Je-li term konstatní symbol $a \in \mathsf{Kons}$, pak jeho hodnota je prvek $[\![a]\!]_\rho = [\![a]\!]$. Je-li term proměnná x, pak jeho hodnota je $[\![x]\!]_\rho = \rho(x)$.
2. Je-li $f(t_1, \ldots, t_n)$ term. pak jeho hodnota je

   $$[\![f(t_1, \ldots, t_n)]\!]_\rho = [\![f]\!]([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho).$$

   [Jinými slovy, hodnota termu $f(t_1, \ldots, t_n)$ je funkční hodnota funkce $[\![f]\!]$ provedené na n-tici prvků $[\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho$ z U.]

Poznamenejme, že neobsahuje-li term t proměnnou, pak jeho hodnota nezáleží na kontextu proměnných ρ, ale pouze na interpretaci.

Tuto formální definici si můžete přiblížit ještě takto. Vezmeme term t a utvoříme jeho syntaktický strom. Listy stromu ohodnotíme tak, jak nám říká interpretace (pro konstantní symboly) a kontext proměnných (pro proměnné). Pak jdeme v syntaktickém stromu směrem ke kořeni. Vrchol, který odpovídá n-árnímu funkčnímu symbolu f a má následníky ohodnoceny prvky d_1, d_2, ..., d_n (v tomto pořadí zleva doprava), ohodnotíme prvkem $[\![f]\!](d_1, \ldots, d_n)$, tj. obrazem n-tice $(d_1, \ldots, d_n)$ v zobrazení $[\![f]\!]$. Prvek, kterým je ohodnocen kořen, je hodnota celého termu v dané interpretaci a daném kontextu. Uvědomte si, že se jedná o přesně stejný postup jako např. při vyhodnocování algebraických výrazů.

2.2.4 Pravdivostní hodnota formule v dané interpretaci a daném kontextu. Nejprve definujeme *pravdivost formulí v dané interpretaci* $\langle U, [\![-]\!]\rangle$ při daném kontextu proměnných ρ:

1. Nechť φ je atomická formule. Tj. $\varphi = P(t_1, \ldots, t_n)$, kde P je predikátový symbol arity n a $t_1, \ldots, t_n$ jsou termy. Pak φ je pravdivá v interpretaci $\langle U, [\![-]\!]\rangle$ a kontextu ρ právě tehdy, když

   $$([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho) \in [\![P]\!].$$

Jinými slovy: φ je v naší interpretaci pravdivá právě tehdy, když n-tice hodnot termů $([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho)$ má vlastnost $[\![P]\!]$.

2. Jsou-li φ a ψ formule, jejichž pravdivost v interpretaci $\langle U, [\![-]\!]\rangle$ a kontextu ρ již známe, pak

   - $\neg\varphi$ je pravdivá právě tehdy, když φ není pravdivá.
   - $\varphi \wedge \psi$ je pravdivá právě tehdy, když φ i ψ jsou pravdivé.
   - $\varphi \vee \psi$ je nepravdivá právě tehdy, když φ i ψ jsou nepravdivé.
   - $\varphi \Rightarrow \psi$ je nepravdivá právě tehdy, když φ je pravdivá a ψ je nepravdivá.
   - $\varphi \Leftrightarrow \psi$ je pravdivá právě tehdy, když buď obě formule φ a ψ jsou pravdivé, nebo obě formule φ a ψ jsou nepravdivé.

3. Je-li φ formule a x proměnná, pak

   - $\forall x \varphi(x)$ je pravdivá právě tehdy, když fromule φ je pravdivá *v každém* kontextu $\rho[x := d]$, kde d je prvek U.
   - $\exists x\, \varphi(x)$ je pravdivá právě tehdy, když fromule φ je pravdivá *v aspoň jednom* kontextu $\rho[x := d]$, kde d je prvek U..

2.2.5 Pravdivostní hodnota sentence. Sentence φ je *pravdivá v interpretaci* $\langle U, [\![-]\!]\rangle$ právě tehdy, když je pravdivá v každém kontextu proměnných ρ.

Poznamenejme, že pro sentence v předchozí definici jsme mohli požadovat pravdivost v alespoň jednom kontextu.

2.2.6 Model sentence. Interpretace $\langle U, [\![-]\!]\rangle$, ve které je sentence φ pravdivá, se nazývá *model sentence* φ.

2.2.7 Splnitelné množiny sentencí. Množina sentencí M je *splnitelná* právě tehdy, když existuje interpretace $\langle U, [\![-]\!]\rangle$, v níž jsou všechny sentence z M pravdivé. Takové interpretaci pak říkáme *model* množiny sentencí M.

Množina sentencí M je *nesplnitelná*, jestliže ke každé interpretaci $\langle U, [\![-]\!]\rangle$ existuje formule z M, která je v $\langle U, [\![-]\!]\rangle$ nepravdivá.

Z poslední definice vyplývá, že prázdná množina sentencí je splnitelná. (Porovnejte s výrokovou logikou.)

2.3 Tautologická ekvivalence

2.3.1 Tautologická ekvivalence sentencí. Řekneme, že dvě sentence φ a ψ jsou *tautologicky ekvivalentní* právě tehdy, když mají stejné modely, tj. jsou pravdivé ve stejných interpretacích. Jinými slovy, mají stejnou pravdivostní hodnotu ve všech interpretacích.

Někdy se říká, že sentence jsou *sémanticky* ekvivalentní místo, že jsou tautologicky ekvivalentní.

2.3.2 Poznámka. Dá se jednoduše dokázat, že tautologická ekvivalence je relace ekvivalence na množině všech sentencí daného jazyka $\mathcal{L}$ a že má podobné vlastnosti jako tautologická ekvivalence formulí výrokové logiky.

2.3.3 Tvrzení. Nechť φ a ψ jsou sentence. Pak platí:

$$\varphi \models\!\models \psi \quad \text{právě tehdy, když} \quad \varphi \Leftrightarrow \psi \text{ je tautologie.}$$

2.3.4 Poznámka. Ze známých tautologií dostáváme následující tautologické ekvivalence

1. $\neg(\forall x\, P(x)) \models\!\models (\exists x\, \neg P(x))$,
2. $\neg(\exists x\, P(x)) \models\!\models (\forall x\, \neg P(x))$,
3. $\forall x\, \forall y\, Q(x,y) \models\!\models \forall y\, \forall x\, Q(x,y)$,
4. $\exists x\, \exists y\, Q(x,y) \models\!\models \exists y\, \exists x\, Q(x,y)$.

2.3.5 Tvrzení. Platí následující tautologické ekvivalence (P a Q jsou unární predikátové symboly).

1. $(\forall x\, P(x)) \wedge (\forall x\, Q(x)) \models\!\models \forall x\, (P(x) \wedge Q(x))$;
2. $(\exists x\, P(x)) \vee (\exists x\, Q(x)) \models\!\models \exists x\, (P(x) \vee Q(x))$;
3. $(\forall x\, P(x)) \vee (\forall y\, Q(y)) \models\!\models \forall x\, \forall y\, (P(x) \vee Q(y))$;
4. $(\exists x\, P(x)) \wedge (\exists y\, Q(y)) \models\!\models \exists x\, \exists y\, (P(x) \wedge Q(y))$.

2.4 Sémantický důsledek

Obdobně jako ve výrokové logice definujeme i v predikátové logice pojem *sémantický důsledek* (též *konsekvent*, *tautologický důsledek*); tentokrát však jen pro množiny sentencí.

2.4.1 Sémantický důsledek. Řekneme, že sentence φ je *sémantickým důsledkem*, též *konsekventem* množiny sentencí S právě tehdy, když každý model množiny S je také modelem sentence φ. Tento fakt značíme

$$S \models \varphi.$$

Můžeme též říci, že sentence φ *není* konsekventem množiny sentencí S, jestliže existuje model množiny S, který není modelem sentence φ. To znamená, že existuje interpretace $\langle U, [\![-]\!]\rangle$, v níž je pravdivá každá sentence z množiny S a není pravdivá formule φ. Jedná se tedy o obdobný pojem jako ve výrokové logice, pouze místo o pravdivostním ohodnocení mluvíme o interpretaci.

2.4.2 Konvence. Jestliže množina sentencí S je jednoprvková, tj. $S = \{\psi\}$, pak píšeme $\psi \models \varphi$ místo $\{\psi\} \models \varphi$. Je-li množina S prázdná, píšeme $\models \varphi$ místo $\emptyset \models \varphi$.

2.4.3 Příklady. Jsou-li P a Q unární predikátové symboly a R binární predikátový symbol, pak

1. $\forall x\, P(x) \models \exists x\, P(x)$.

   Ano, jestliže všechny objekty mají vlastnost odpovídající P, pak existuje alesponň jeden objekt, který tuto vlastnost má. Uvědomte si, že $\exists x\, P(x)$ je sémantickým důsledkem sentence $\forall x\, P(x)$ z toho důvodu, že domain U v interpretaci nikdy není prázdný.

2. $\exists x\, P(x) \not\models \forall x\, P(x)$.

   Není těžké najít interpretaci $\langle U, [\![-]\!]\rangle$, ve které je pravdivá sentence $\exists x\, P(x)$ a sentence $\forall x\, P(x)$ je nepravdivá. Položme např. $U = \mathbf{N}$, tj. naše objekty jsou přirozená čísla, a vlastnost P je vlastnost býti sudým číslem, tj. $[\![P]\!]$ je množina sudých přirozených čísel.

3. $\{\forall x\,(P(x) \Rightarrow Q(x)), \exists x\, P(x)\} \models \exists x\, Q(x)$.

   Ano, jestliže v nějaké interpretaci existuje objekt, který má vlastnost odpovídající P, pak z pravdivosti implikace $\forall x\,(P(x) \Rightarrow Q(x))$ tento objekt musí mít vlastnost odpovídající Q.

4. $(\forall x\, P(x)) \vee (\forall x\, Q(x)) \models \forall x\,(P(x) \vee Q(x))$.

   Ano, sentence $(\forall x\, P(x)) \vee (\forall x\, Q(x))$ je stejná jako sentence $(\forall x\, P(x)) \vee (\forall y\, Q(y))$ (přejmenovali jsme pouze v druhé části sentence vázanou proměnnou x na y). Tedy sentence $(\forall x\, P(x)) \vee (\forall x\, Q(x))$ je pravdivá v každé interpretaci, ve které všechny objekty mají vlastnost odpovídající P nebo všechny objekty mají vlastnost odpovídajicí Q (nebo obě vlastnosti). To ale znamená, že každý z objektů má aspoň jednu z vlastností odpovídající P a Q. Tedy sentence $\forall x\,(P(x) \vee Q(x))$ je pravdivá.

5. $(\forall x\,(P(x) \vee Q(x)) \not\models (\forall x\,(P(x)) \vee (\forall x\, Q(x))$.

   Není těžké najít interpretaci, ve které je pravdivá sentence $\forall x\,(P(x) \vee Q(x))$, tj. každý objekt této interpretace má aspoň jednu z vlastností odpovídajících P a Q, a přitom některé objekty mají jen P a některé jen Q. Tudíž není pravdivé, že všechny objekty mají vlastnost odpovídající

P nebo všechny objekty mají vlastnost odpovídající Q; tudíž sentence $(\forall x\, P(x)) \vee (\forall x\, Q(x))$ je nepravdivá.

Uvažujme interpretaci, kde U je množina přirozených čísel, vlastnost odpovídající P je být sudým číslem, vlastnost odpovídající Q je být lichým číslem. Pak platí, že každé přirozené číslo je sudé nebo liché, ale neplatí, že by všechna přirozená čísla byla sudá nebo že by všechna přirozená čísla byla lichá.

6. $(\forall x\, P(x)) \vee (\forall y\, Q(y)) \models \forall x\, \forall y\, (P(x) \vee Q(y))$.

   Nejprve označme $\alpha = (\forall x\, P(x)) \vee (\forall y\, Q(y))$ a $\beta = \forall x\, \forall y\, (P(x) \vee Q(y))$.

   Toto tvrzení je trochu obtížnější zdůvodnit. Půjdeme na to tak, že ukážeme, že není-li sentence β pravdivá, není pravdivá ani sentence α. Tím bude 5) dokázáno.

   Fakt, že sentence β je pravdivá v interpretaci $\langle U, [\![-]\!]\rangle$ znamená toto: vybereme-li libovolnou dvojici objektů $c, d \in U$, pak objekt c má vlastnost $[\![P]\!]$ nebo objekt d má vlastnost $[\![Q]\!]$. Předpokládejme tedy, že sentence β není pravdivá v interpretaci $\langle U, [\![-]\!]\rangle$. To znamená, že máme (aspoň) jednu dvojici $c, d \in U$ takovou, že c nemá $[\![P]\!]$ a d nemá $[\![Q]\!]$. Pak ale nemůže být pravdivá ani sentence $\forall x\, P(x)$ (c nemá vlastnost $[\![P]\!]$), ani sentence $\forall y\, Q(y)$ (d nemá vlastnost $[\![Q]\!]$). Proto je nepravdivá i sentence α.

7. $\exists x\, \forall y\, R(x, y) \models \forall y\, \exists x\, R(x, y)$.

   Ano, sentence $\exists x\, \forall y\, R(x, y)$ je pravdivá ve všech interpretacích, ve kterých existuje prvek $c \in U$ takový, že c s každým prvkem $d \in U$ má vlastnost odpovídající R (nepřesně, ale srozumitelně, můžeme psát, že $R(c, d)$ je pravdivé). Pak je však pravdivá i sentence $\forall y\, \exists x\, R(x, y)$, protože pro libovolné $d \in U$ můžeme do proměnné x dosadit prvek c.

8. $\forall x\, \exists y\, R(x, y) \not\models \exists y\, \forall x\, R(x, y)$.

   Zde najdeme lehce interpretaci, ve které je sentence $\forall y\, \exists x\, R(x, y)$ pravdivá, ale sentence $\forall x\, \exists y\, R(x, y)$ nikoli:

   Jako domain U zvolíme množinu všech přirozených čísel a predikát $R(x, y)$ interpretujeme jako $x < y$. (Tj. $[\![R]\!]$ je množina všech uspořádaných dvojic (m, n), kde $m < n$.) Pak formule $\forall x\, \exists y\, R(x, y)$ je pravdivá, protože opravdu pro každé přirozené číslo n existuje číslo větší, např. $n + 1$. Sentence $\exists y\, \forall x\, R(x, y)$ pravdivá v této interpretaci není; to by znamenalo, že bychom museli mít největší přirozené číslo a takové číslo neexistuje.

2.4.4 Obdobně jako pro výrokovou logiku, dostáváme řadu jednoduchých prozorování. Pro množiny sentencí M, N a sentenci φ platí:

1. Je-li $\varphi \in M$, je $M \models \varphi$.
2. Je-li $N \subseteq M$ a $N \models \varphi$, je i $M \models \varphi$.
3. Je-li φ tautologie, pak $M \models \varphi$ pro každou množinu sentencí M.
4. Je-li $\models \varphi$, pak φ je tautologie.
5. Je-li M nesplnitelná množina, pak $M \models \varphi$ pro každou sentenci φ.

2.4.5 Tvrzení. Nechť φ a ψ jsou sentence. Pak platí:

$$\varphi \models\!\models \psi \quad \text{právě tehdy, když} \quad \varphi \models \psi \text{ a } \psi \models \varphi.$$

2.4.6 Tvrzení. Nechť φ a ψ jsou sentence. Pak platí:

$$\varphi \models \psi \quad \text{právě tehdy, když} \quad \varphi \Rightarrow \psi \text{ je tautologie.}$$

2.4.7 Věta. Pro každou množinu sentencí S a každou sentenci φ platí:

$$S \models \varphi \quad \text{právě tehdy, když} \quad S \cup \{\neg\varphi\} \quad \text{je nesplnitelná množina.}$$

2.5 Rezoluční metoda v predikátové logice

Rezoluční metoda v predikátové logice je obdobná stejnojmenné metodě ve výrokové logice. Ovšem vzhledem k bohatší vnitřní struktuře formulí predikátové logiky je složitější. Používá se v logickém programování a je základem programovacího jazyka Prolog.

Nejprve zavedeme literály a klausule v predikátové logice.

2.5.1 Literál. *Literál* je atomická formule (tzv. *pozitivní literál*), nebo negace atomické formule (tzv. *negativní literál*). *Komplementární literály* jsou dva literály, z nichž jeden je negací druhého.

2.5.2 Klausule *Klausule* je sentence taková, že všechny kvantifikátory jsou obecné a stojí na začátku sentence (na jejich pořadí nezáleží) a za nimi následují literál nebo disjunkce literálů.

Ve výrokové logice jsme pro každou formuli α našli k ní tautogicky ekvivalentní množinu klausulí S_α a to tak, že α i S_α byly pravdivé ve stejných pravdivostních ohodnocení. Takto jednoduchá situace v predikátové logice není. V příští přednášce si ukážeme, jak k dané sentenci φ najít množinu klausulí S_φ a to tak, že φ je splnitelná právě tehdy, když množina S_φ je splnitelná.

2.5.3 Převedení sentence na klausální tvar. Pro každou sentenci φ existuje množina klausulí S_φ taková, že sentence φ je splnitelná právě tehdy, když S_φ je splnitelná.

2.5.4 Postup. Uvedeme postup, kterým pro danou sentenci φ zkonstruujeme množinu klausulí S_φ.

1. Přejmenujeme proměnné formule φ tak, aby každý vstup kvantifikátoru vázal jinou proměnnou. (Tj. např. formuli $\forall x\, P(x) \vee \forall x\, Q(x, a)$ nahradíme formulí $\forall x\, P(x) \vee \forall y\, Q(y, a)$.)

2. Spojky $\Rightarrow$, $\Leftrightarrow$ nahradíme spojkami $\neg$, $\vee$ a $\wedge$ na základě známých tautologických rovností

$$\begin{aligned} \alpha \Rightarrow \beta \quad &\text{nahradíme} \quad \neg\alpha \vee \beta \\ \alpha \Leftrightarrow \beta \quad &\text{nahradíme} \quad (\neg\alpha \vee \beta) \wedge (\alpha \vee \neg\beta) \end{aligned}$$

3. Přesuneme spojku $\neg$ „co nejníže“ v derivačním stromu formule, tj. až před atomické formule. Použijeme k tomu vztahy

$$\begin{array}{rll} \neg\exists x\,\alpha & \text{nahradíme} & \forall x\,\neg\alpha \\ \neg\forall x\,\alpha & \text{nahradíme} & \exists x\,\neg\alpha \\ \neg(\alpha\vee\beta) & \text{nahradíme} & \neg\alpha\wedge\neg\beta \\ \neg(\alpha\wedge\beta) & \text{nahradíme} & \neg\alpha\vee\neg\beta \\ \neg\neg\alpha & \text{nahradíme} & \alpha \end{array}$$

4. Přesuneme spojku $\vee$ „co nejníže“ v derivačním stromu formule pomocí vztahů

$$\begin{array}{rll} \alpha\vee(\beta\wedge\gamma) & \text{nahradíme} & (\alpha\vee\beta)\wedge(\alpha\vee\gamma) \\ \alpha\vee(\forall x\,\beta) & \text{nahradíme} & \forall x\,(\alpha\vee\beta) \\ \alpha\vee(\exists x\,\beta) & \text{nahradíme} & \exists x\,(\alpha\vee\beta) \end{array}$$

Přitom dáváme přednost první rovnosti. Teprve v případě, že první rovnost nelze aplikovat, používáme další dvě rovnosti. Uvědomte si, že druhá rovnost je opravdu tautologická ekvivalence pouze proto, že formule α neobsahuje proměnnou x (viz krok 1).

5. Použijeme tautologickou ekvivalenci

$$\forall x\,(\alpha\wedge\beta) \quad \text{nahradíme} \quad (\forall x\,\alpha)\wedge(\forall x\,\beta)$$

k distribuci obecného kvantifikátoru. Jestliže nyní formule neobsahuje existenční kvantifikátor, máme formuli ψ, která je konjunkcí klausulí. Tuto formuli tedy nahradíme množinou S_φ jejích klausulí.
V případě, že formule ψ obsahuje existenční kvantifikátor, provedeme skolemisaci, která je vysvětlena a popsána v dalších odstavcích.

2.5.5 Skolemizace. Poznamenejme, že termíny „skolemizace“, „skolemizační konstanta“ a „skolemizační funkční symbol“ jsou odvozeny od jména norského matematika — logika Thoralfa Skolema.

Skolemizací nahradíme formuli ψ formulí ψ' takovou, že formule ψ je splnitelná právě tehdy, když je splnitelná formule ψ' (obecně ale ne pro stejnou interpretaci). Dříve než ukážeme obecný postup, uvedeme dva příklady.

2.5.6 Příklad 1. Najděme klausuli ψ', která je splnitelná právě tehdy, když je splnitelná sentence $\psi = \exists x\, P(x)$.

Hledanou sentencí je $\psi' = P(a)$, kde a je nějaký nový konstantní symbol. Není obtížné nahlédnout, že formule ψ je tautologickým důsledkem formule ψ'.

Konstantní symbol a použitý v minulém příkladě, se nazývá *skolemizační konstanta*.

2.5.7 Příklad 2. Najděme klausuli ψ', která je splnitelná právě tehdy, když je splnitelná sentence $\psi = \forall x\, \exists y\, Q(x, y)$.

Formuli ψ nahradíme formulí $\psi' = \forall x Q(x, f(x))$, kde f je nový unární funkční symbol. Nyní již opět platí, že formule ψ je splnitelná právě tehdy, když je splnitelná formule ψ'.

Funkčnímu symbolu f se říká *skolemizační funkční symbol*.

2.5.8 Obecný postup (pokračování bodů 1 – 5 z postupu 2.5.4).

6. Obsahuje-li formule existenční kvantifikátor, nahradíme každou uzavřenou podformuli tvaru $\forall x_1 \ldots \forall x_n\, \exists y\, \alpha(y)$ formulí $\forall x_1 \ldots \forall x_n\, \alpha(f(x_1, \ldots, x_n))$, kde f je libovolný *nový* funkční symbol arity n. Je-li $n = 0$, použijeme *nový* konstantní symbol a. Tomuto procesu se říká *skolemizace*, funkčnímu symbolu f *skolemizační funkční symbol*, konstantě a *skolemizační konstanta*. Pokračujeme podle kroku 5 z 2.5.4.

Uvědomte si, že proměnné $x_1, \ldots, x_n$ jsou právě všechny proměnné vázané obecným kvantifikátorem, na které narazíme při postupu syntaktickým stromem od $\exists y$ směrem ke kořeni.

2.5.9 Rezolventy klausulí. Ve výrokové logice jsme rezolventy vytvářeli tak, že jsme si vždy vzali dvě klausule, které obsahovaly dvojici komplementárních literálů, a výsledná rezolventa byla disjunkcí všech ostatních literálů z obou klausulí. Situace v predikátové logice je složitější. Postup, jak vytváříme rezolventy v predikátové logice, si ukážeme na příkladech.

2.5.10 Příklad. Najděme rezolventu klausulí $K_1 = \forall x\, \forall y\, (P(x) \vee \neg Q(x, y))$ a $K_2 = \forall x\, \forall y\, (Q(x, y) \vee R(y))$, kde P a R jsou unární predikátové symboly a Q je binární predikátový symbol, x, y jsou proměnné.

Klausule K_1 a K_2 obsahují dvojici komplementárních literálů, totiž $\neg Q(x, y)$ je literál K_1 a $Q(x, y)$ je literál K_2. Rezolventou klausulí K_1 a K_2 je tedy $K = \forall x\, \forall y\, (P(x) \vee R(y))$.

2.5.11 Příklad. Najděme rezolventu klausulí $K_1 = \forall x\, (P(x) \vee \neg Q(x))$ a $K_2 = Q(a) \vee R(b)$, kde P, Q jsou unární predikátové symboly a a, b jsou konstanty.

Klausule K_1 a K_2 neobsahují dvojici komplementárních literálů, neboť negace literálu $Q(a)$ není literál $\neg Q(x)$ a naopak. Přitom však literál $\neg Q(x)$ odpovídá formuli $\forall x\, \neg Q(x)$, a tedy zahrnuje i $\neg Q(a)$. Proto při substituci konstanty a za proměnnou x dostáváme klausule

$$K_1' = P(a) \vee \neg Q(a) \quad \text{a} \quad K_2' = Q(a) \vee R(b)$$

a jejich rezolventa je klausule $P(a) \vee R(b)$.

2.5.12 Příklad. Najděme rezolventu klausulí

$$K_1 = \forall x\, \forall y\, (\neg P(x,y) \vee Q(x,y,a)) \quad \text{a} \quad K_2 = \forall z\, \forall v\, (\neg Q(g(v),z,z) \vee R(v,z)),$$

kde P, R jsou binární predikátové symboly, Q je predikátový symbol arity 3 a a je konstanta.

Pokusíme se vhodnou substitucí vytvořit z $Q(x,y,a)$ a $\neg Q(g(v),z,z)$ dvojici komplementárních literálů. Toho dosáhneme substitucí: za proměnné y a z dosadíme konstantu a, a za proměnnou x dosadíme term $g(v)$. Tím dostaneme komplementární literály

$$Q(g(v),a,a) \qquad \neg Q(g(v),a,a).$$

Provedením substituce na celé klausule, dostaneme klausule K_1' a K_2', jejichž rezolventa bude i rezolventou klausulí K_1, K_2. Tedy $K_1' = \forall v\, \neg(P(g(v),a) \vee Q(g(v),a,a))$, $K_2' = \forall v\, (\neg Q(g(v),a,a) \vee R(v,a))$ a hledaná rezolventa je $K = \forall v\, (\neg P(g(v),a) \vee R(v,a))$.

2.5.13 Poznámka. Ne vždy rezolventa existuje. Rezolventa klausulí

$$K_1 = \forall x\, (\neg P(x) \vee Q(f(x),a)) \quad \text{a} \quad K_2 = \forall y\, \forall z\, (\neg Q(y,y) \vee R(f(y),z)),$$

(kde P je unární predikátový symbol, Q a R jsou binární predikátové symboly, f je unární funkční symbol a a je konstanta) neexistuje. Případy, kdy existuje či neexistuje vhodná substituce, řeší unifikační algoritmus, který uvádíme v následujícícm odstavci. V případě, že substituce existuje, unifikační algoritmus najde nejobecnější substituci.

2.5.14 Unifikační algoritmus.

Vstup: Dva positivní literály L_1, L_2, které nemají společné proměnné.
Výstup: Hlášení `neexistuje` v případě, že hledaná substituce neexistuje,
v opačném případě substituce ve tvaru množiny prvků tvaru x/t,
kde x je proměnná, za kterou se dosazuje,
a t je term, který se za proměnnou x dosazuje.

1. Položme $E_1 := L_1$, $E_2 := L_2$, $\theta := \emptyset$.
2. Jsou-li E_1, E_2 prázdné řetězce, stop. Množina θ určuje hledanou substituci. V opačném případě položíme $E_1 := E_1\theta$, $E_2 := E_2\theta$ (tj. na E_1, E_2 provedeme substituci θ).
3. Označíme X první symbol řetězce E_1, Y první symbol řetězce E_2.

4. Je-li $X = Y$, odstraníme X a Y z počátku E_1 a E_2. Jsou-li X a Y predikátové nebo funkční symboly, odstraníme i jim příslušné závorky a jdeme na krok 2.

5. Je-li X proměnná, neděláme nic.
   Je-li Y proměnná (a X nikoli), přehodíme E_1, E_2 a X, Y.
   Není-li ani X ani Y proměnná, stop. Výstup `neexistuje`.

6. Je-li Y proměnná nebo konstanta, položíme $\theta := \theta \cup \{X/Y\}$. Odstraníme X a Y ze začátků řetězců E_1 a E_2 (spolu s čárkami, je-li třeba) a jdeme na krok 2.

7. Je-li Y funkční symbol, označíme Z výraz skládající se z Y a všech jeho argumentů (včetně závorek a čárek). Jestliže Z obsahuje X, stop, výstup `neexistuje`.
   V opačném případě položíme $\theta := \theta \cup \{X/Z\}$, odstraníme X a Z ze začátků E_1 a E_2 (odstraníme čárky, je-li třeba) a jdeme na krok 2.

2.5.15 Rezoluční princip. Je obdobný jako rezoluční princip ve výrokové logice:

Je dána množina klausulí S. Označme

$$\begin{aligned} R(S) &= S \cup \{K \mid K \text{ je nejobecnější rezolventa některých klausulí z } S\} \\ R^0(S) &= S \\ R^{i+1}(S) &= R(R^i(S)) \quad \text{pro} \quad i \in \mathbb{N} \\ R^\star(S) &= \bigcup \{R^i(S) \mid i \geq 0\}. \end{aligned}$$

Množina klausulí S je splnitelná právě tehdy, když $R^\star(S)$ neobsahuje prázdnou klausuli F.

Jestliže je množina S konečná, existuje přirozené číslo n_0 takové, že $R^{n_0}(S) = R^{n_0 1}$. Pak $R^\star(S) = R^{n_0}(S)$.

2.5.16 Poznámka. Rezoluční princip používáme i pro zjištění, zda z množiny sentencí S vyplývá sentence α (tj. zda $S \models \alpha$) a to takto.

Postupujeme podle věty 2.4.7: **Nejprve** vytvoříme množinu $S \cup \{\neg\alpha\}$ a pak upravujeme sentence této množiny na klausální tvar; množinu, kterou dostaneme, označme M. Jestliže $R^\star(M)$ obsahuje prázdnou klausuli, pak množina $S \cup \{\neg\alpha\}$ je nesplnitelná a $S \models \alpha$ platí; jestliže prázdná klausule nepatří do $R^\star(M)$, pak $S \cup \{\neg\alpha\}$ je splnitelná a $S \models \alpha$ neplatí.

2.5.17 Příklad 1. Je dána množina sentencí

$$S = \{\forall x\, \forall y\, ((P(x) \wedge Q(x,y)) \Rightarrow R(y)), \forall x\, Q(f(x), g(x)), P(f(b))\}$$

a sentence $\varphi = R(g(b))$, kde P je unární predikátový symbol, Q je binární predikátový symbol, f a g jsou binární funkční symboly a b je konstantní symbol.

Rezoluční metodou rozhodněte, zda platí $S \models \varphi$.

2.5.18 Řešení. Nejprve vytvoříme množinu $S \cup \{\neg\varphi\}$. Máme

$$S\cup\{\neg\varphi\} = \{\forall x\,\forall y\,((P(x)\wedge Q(x,y)) \Rightarrow R(y)), \forall x\,Q(f(x),g(x)), P(f(b)), \neg R(g(b))\}.$$

Nyní převedeme sentence na klausální tvar. Dostaneme množinu klausulí

$$\{\forall x\,\forall y\,(\neg P(x) \vee \neg Q(x,y) \vee R(y)), \forall z\,Q(f(z),g(z)), P(f(b)), \neg R(g(b))\}.$$

Označme jednotlivé klausule: $K_1 = \forall x\,\forall y\,(\neg P(x) \vee \neg Q(x,y) \vee R(y))$, $K_2 = \forall z\,Q(f(z),g(z))$, $K_3 = P(f(b))$ a $K_4 = \neg R(g(b))$.

Nejprve utvoříme rezolventu klausulí K_1 a K_3. Abychom rezolventu mohli utvořit, za proměnnou x dosadíme term $f(b)$. Dostaneme klausuli $K_1' = \forall y\,(\neg P(f(b)) \vee \neg Q(f(b),y) \vee R(y))$. Klausule K_1' a K_3 obsahují dvojici komplementárních literálů, totiž $\neg P(f(b))$ v K_1' a $P(f(b))$ v K_3. Jejich rezolventa je klausule $K_5 = \forall y\,(\neg Q(f(b),y) \vee R(y))$.

Nyní utvoříme rezolventu klausulí K_4 a K_5. Provedeme substituci: za proměnnou y dosadíme term $f(b)$. Dostaneme klausule $K_5' = \neg Q(f(b),g(b)) \vee R(g(b))$ a $K_4 = \neg R(g(b))$. Jejich rezolventa je klausule $K_6 = \neg Q(f(b),g(b))$.

Nyní vybereme klausule K_2 a K_6. Po dosazení konstanty b za proměnnou z, dostáváme klausule $K_2' = Q(f(b),g(b))$ a $K_6 = \neg Q(f(b),g(b))$, jejichž rezolventa je prázdná klausule F.

Proto je množina $S \cup \{\neg\varphi\}$ nesplnitelná a $S \models \varphi$ platí.

Poznamenejme, že jsme nekonstruovat celou množinu $R^\star(S \cup \{\neg\varphi\})$; nebylo to potřeba proto, že prázdnou klausuli jsme dostali v $R^3(S \cup \{\neg\varphi\})$.

2.5.19 Poznámka. Pomocí rezoluční metody můžeme také zjišťovat, zda dvě sentence jsou nebo nejsou tautologicky ekvivalentní. Ukažme si to na příkladě.

2.5.20 Příklad 2. Pro dvě sentence φ a ψ rezoluční metodou rozhodněte, zda $\varphi \models\!\models \psi$, kde $\varphi = \exists x\,\forall y\,Q(x,y)$ a $\psi = \forall y\,\exists x\,Q(x,y)$.

2.5.21 Řešení. Abychom ověřili, že $\varphi \models\!\models \psi$, musíme ukázat:

$$\varphi \models \psi \quad \text{a} \quad \psi \models \varphi.$$

Add 1. Utvoříme množinu $\{\varphi, \neg\psi\}$. Této množině odpovídá množina sentencí $\{\exists x\,\forall y\,Q(x,y), \exists t\,\forall z\,\neg Q(z,t)\}$. Skolemizujeme a dostaneme množinu klausulí $S = \{\forall y\,Q(a,y), \forall z\,\neg Q(z,b)\}$. Substitucí y/b a z/a, dostaneme dvojici klausulí $K_1 = Q(a,b)$ a $K_2 = \neg Q(a,b)$; jejich rezolventa je prázdná klausule. Proto $\varphi \models \psi$ platí.

Add 2. Obdobně utvoříme množinu $\{\psi, \neg\varphi\}$. Této množině odpovídá množina sentencí $\{\forall x\,\exists y\,Q(x,y), \forall z\,\exists t\,\neg Q(z,t)\}$. Skolemizujeme a dostaneme množinu klausulí $S = \{\forall x\,Q(x,f(x)), \forall z\,\neg Q(z,g(z))\}$. Nyní sice můžeme provést substituci z/x, ale dostáváme dvě klausule $K_1 = \forall x\,Q(x,f(x))$ a $K_2 = \forall x\,\neg Q(x,g(x))$, jejichž rezolventa neexistuje. Proto $\psi \models \varphi$ neplatí a neplatí proto ani $\varphi \models\!\models \psi$.

2.6 Přirozená dedukce

V předchozích přednáškách jsme se zabývali sémantickým důsledkem jak ve výrokové, tak i predikátové logice. Matematická logika se zabývá také tzv. logickým důsledkem. Formule α je logickým důsledkem množiny formulí S, jestliže je možno ji odvodit z S v některém odvozovacím systému. V následujícím přiblížíme odvozovací systém, který se nazývá přirozená dedukce.

2.6.1 Přirozená dedukce ve výrokové logice. Systém odvozovacích pravidel přirozené dedukce obsahuje pro každou logickou spojku $\neg$, $\vee$, $\wedge$ a $\Rightarrow$ dvě pravidla. Jedno pravidlo spojku „zavádí“, tj. umožňuje napsat formuli s touto spojkou, druhé spojku „odstraňuje“. Prvním pravidlům říkáme *I-pravidla* („I“ pro introdukci, tj. zavedení), druhým *E-pravidla* („E“ pro eliminaci, neboli odstranění) pro danou spojku.

Pravidla jsou tato :

I-pravidlo pro $\wedge$:
$$\dfrac{\varphi \quad \psi}{\varphi \wedge \psi \quad \psi \wedge \varphi}$$

E-pravidlo pro $\wedge$:
$$\dfrac{\varphi \wedge \psi}{\varphi \quad \psi}$$

E-pravidlo pro $\Rightarrow$ (také zvané Modus Ponens):
$$\dfrac{\varphi \quad \varphi \Rightarrow \psi}{\psi}$$

I-pravidlo pro $\vee$:
$$\dfrac{\varphi}{\varphi \vee \psi \quad \psi \vee \varphi}$$

E-pravidlo pro $\neg$:
$$\dfrac{\neg\neg\varphi}{\varphi}$$

I-pravidlo pro $\Rightarrow$:
$$\dfrac{\begin{array}{|l} \varphi \\ \hline \phi \end{array}}{\varphi \Rightarrow \phi}$$

Při vlastním používání budeme říkat, že φ je pomocný předpoklad *aktivní* uvnitř celého pododvození. Dospějeme-li k závěru (tj. formuli $\varphi \Rightarrow \phi$), řekneme, že jsme tento pomocný předpoklad *eliminovali* a on se stal *pasivním* pomocným předpokladem.

E-pravidlo pro $\vee$:
$$\dfrac{\varphi \vee \psi \quad \begin{array}{|l} \varphi \\ \hline \alpha \end{array} \quad \begin{array}{|l} \psi \\ \hline \alpha \end{array}}{\alpha}$$

I-pravidlo pro $\neg$:
$$\dfrac{\begin{array}{|l} \varphi \\ \hline \alpha \\ \neg\alpha \end{array}}{\neg\varphi}$$

2.6.2 Odvození. Posloupnost formulí φ_1, φ_2, ..., φ_n se nazývá *odvození z předpokladů* S právě tehdy, když

- každá formule φ_i je buď předpoklad (tj. $\varphi_i \in S$), nebo je pomocný předpoklad, nebo vznikla z předcházejících formulí pomocí některého odvozovacího pravidla;

- všechny pomocné předpoklady jsou již pasivní.

2.6.3 Logický důsledek. Formule φ je *logický důsledek množiny předpokladů* S, též *logicky vyplývá z* S, právě tehdy, když existuje odvození z S takové, že $\varphi_n = \varphi$. Zapisujeme $S \vdash \varphi$.

2.6.4 Věta o úplnosti. Pro každou množinu formulí S a formuli φ platí

$$S \vdash \varphi \quad \text{právě tehdy, když} \quad S \models \varphi.$$

2.6.5 Přirozená dedukce v predikátové logice. Kromě odvozovacích pravidel výrokové logiky máme zde ještě dvě odvozovací pravidla pro každý z kvantifikátorů $\forall$ a $\exists$.

E-pravidlo pro $\forall$:
$$\frac{\forall x\, \phi(x)}{\phi(t)}$$

I-pravidlo pro $\forall$:
$$\frac{\phi(x)}{\forall x\, \phi(x)}$$

I-pravidlo pro $\exists$:
$$\frac{\phi(t)}{\exists x\, \phi(x)}$$

E-pravidlo pro $\exists$:
$$\frac{\exists x\, \phi(x) \qquad \begin{array}{|l} \phi(y) \\ \hline \psi \end{array}}{\psi}$$

I v predikátové logice definujeme odvození a logický důsledek stejně jako v 2.6.2 a 2.6.3. A co je důležité, i v predikátové logice platí věta o úplnosti 2.6.4.

Kapitola 3

Grafy

3.1 Orientované a neorientované grafy

3.1.1 Definice orientovaného grafu. *Orientovaný graf* je trojice $\mathbf{G} = (V, E, \varepsilon)$, kde V je konečná množina *vrcholů* (též zvaných *uzlů*), E je konečná množina jmen *hran* (též zvaných *orientovaných hran*) a ε je přiřazení, které každé hraně $e \in E$ přiřazuje uspořádanou dvojici vrcholů a nazývá se *vztah incidence*.

Jestliže $\varepsilon(e) = (u, v)$ pro $u, v \in V$, říkáme, že vrchol u je *počáteční vrchol* hrany e a vrchol v je *koncový vrchol* hrany e; značíme $PV(e) = u$ a $KV(e) = v$. O vrcholech u, v říkáme, že jsou krajní vrcholy hrany e, též že jsou *incidentní* s hranou e. Jestliže počáteční a koncový vrchol jsou stejné, říkáme, že hrana e je *orientovaná smyčka*.

3.1.2 Definice neorientovaného grafu. *Neorientovaný graf* je trojice $\mathbf{G} = (V, E, \varepsilon)$, kde V je konečná množina *vrcholů* (též zvaných *uzlů*), E je konečná množina jmen *hran* a ε je přiřazení, které každé hraně $e \in E$ přiřazuje množinu $\{u, v\}$ pro vrcholy $u, v \in V$ a nazývá se *vztah incidence*.

Jestliže $\varepsilon(e) = \{u, v\}$ pro $u, v \in V$, říkáme, že u, v jsou *krajní vrcholy* hrany e, též že jsou *incidentní* s hranou e. Je-li $u = v$, říkáme že e je *(neorientovaná) smyčka*.

3.1.3 Paralelní hrany, prostý graf. Jestliže v orientovaném grafu existují dvě různé hrany e_1, e_2 se stejnými počátečními i koncovými vrcholy, tj. $PV(e_1) = PV(e_2)$ a $KV(e_1) = KV(e_2)$, říkáme, že hrany e_1, e_2 jsou *paralelní*.

Jestliže v neorientovaném grafu existují dvě různé hrany e_1, e_2 se stejnými krajními vrcholy, tj. $\varepsilon(e_1) = \varepsilon(e_2)$, říkáme, že hrany e_1, e_2 jsou *paralelní*.

Graf se nazývá *prostý graf*, nemá-li paralelní hrany.

3.1.4 Poznámka. V některé literatuře se temínem *graf* rozumí prostý graf a grafům, ať již orientovaným nebo neorientovaným, které mají paralelní hrany, se říká *multigrafy*. Vzhledem k tomu, že v řadě aplikací hrají paralelní hrany podstatnou roli, my tuto terminologii nebudeme používat.

3.1.5 Stupně vrcholů. Je dán orientovaný graf $G = (V, E, \varepsilon)$. *Vstupní stupeň* vrcholu v, značíme jej $d^-(v)$, je roven počtu hran, pro které je v koncovým

vrcholem, tj.

$$d^-(v) = |\{e \in E; KV(e) = v\}|.$$

Výstupní stupeň vrcholu v, značíme jej $d^+(v)$, je roven počtu hran, pro které je v počátečním vrcholem, tj.

$$d^+(v) = |\{e \in E; PV(e) = v\}|.$$

Stupeň vrcholu v, značíme jej $d(v)$, je roven

$$d(v) = d^-(v) + d^+(v);$$

počtu hran, které jsou incidentní s vrcholem v, kde smyčka je počítána dvakrát.

Je dán neorientovaný graf $G = (V, E, \varepsilon)$. *Stupeň* vrcholu v, značíme jej $d(v)$, je roven počtu hran, které jsou incidentní s vrcholem v, kde smyčka je počítána dvakrát.

3.1.6 Tvrzení. Pro každý graf G (orientovaný nebo neorientovaný) platí

$$\sum_{v \in V} d(v) = 2\,|E|,$$

kde $|E|$ značí počet hran grafu G.

3.1.7 Důsledek. Každý graf má sudý počet vrcholů lichého stupně.

3.1.8 Zadávání grafu. Orientovaný i neorientovaný graf můžeme zadat sezname vrcholů a pro každý vrchol seznamem hran v něm začínajících a/nebo v něm končících. Graf též můžeme zadat maticí sousednosti nebo maticí incidence.

3.1.9 Matice sousednosti. Je dán graf $G = (V, E, \varepsilon)$, kde jsme uspořádalii vrcholy, tj. $V = \{v_1, v_2, \ldots, v_n\}$. Čtvercová matice $\mathbf{M} = (m(i,j))_{i,j=1,\ldots,n}$ se nazývá *matice sousednosti* grafu G, jestliže

- Pro orientovaný graf je $m(i,j)$ počet hran, pro něž je v_i počásteční vrchol a v_j koncový vrchol.
- Pro neorientovaný graf je $m(i,j)$ počet hran s krajními vrcholy v_i a v_j.

Poznamenejme, že pro neorientovaný graf je matice sousednosti symetrická.

3.1.10 Matice incidence. Je dán graf $G = (V, E, \varepsilon)$ bez smyček. Očíslujme vrcholy $V = \{v_1, v_2, \ldots, v_n\}$ a hrany $E = \{e_1, e_2, \ldots, e_m\}$. Matice $\mathbf{B} = (b(i,j))$ typu (n, m) se nazývá *matice incidence* grafu G, jestliže

- Pro orientovaný graf je

$$b(i,j) = \begin{cases} 1, & \text{jestliže } v_i \text{ je počáteční vrchol hrany } e_j, \\ -1, & \text{jestliže } v_i \text{ je koncový vrchol hrany } e_j, \\ 0, & \text{v ostatních případech.} \end{cases}$$

- Pro neorientovaný graf je

$$b(i,j) = \begin{cases} 1, & \text{jestliže } v_i \text{ je krajní vrchol hrany } e_j, \\ 0, & \text{v ostatních případech.} \end{cases}$$

Matice incidence má v každém sloupci jednu 1 a jednu -1 (pro orientované grafy) a dvě 1 (pro neorientované grafy).

3.1.11 Porovnávání grafů. Řekneme, že dva grafy $G = (V, E, \varepsilon)$ a $G' = (V', E', \varepsilon')$ jsou si *rovny*, jestliže $V = V'$, $E = E'$ a $\varepsilon = \varepsilon'$.

Řekneme, že dva grafy $G = (V, E, \varepsilon)$ a $G' = (V', E', \varepsilon')$ jsou *isomorfní*, jestliže existují bijekce $f\colon V \to V'$ a $g\colon E \to E'$ takové, že

$$\varepsilon(e) = (u, v) \quad \text{právě tehdy, když} \quad \varepsilon'(g(e)) = (f(u), f(v))$$

pro orientované grafy a

$$\varepsilon(e) = \{u, v\} \quad \text{právě tehdy, když} \quad \varepsilon'(g(e)) = \{f(u), f(v)\}$$

pro neorientované grafy.

3.1.12 Sled. Je dán orientovaný graf $G = (V, E, \varepsilon)$. *Orientovaný sled* v G je posloupnost vrcholů a hran

$$v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_{k-1}, v_k$$

taková, že pro každé $i = 1, 2, \ldots, k-1$ platí $v_i = PV(e_i)$ a $v_{i+1} = KV(e_i)$.

Neorientovaný sled v G je posloupnost vrcholů a hran

$$v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_{k-1}, v_k$$

taková, že pro každé $i = 1, 2, \ldots, k-1$ platí že vrcholy v_i a v_{i+1} jsou krajní vrcholy hrany e_i.

Neorientovaný sled se od orientovaného sledu liší tím, že můžeme "jít proti směru"hrany.

V neorientovaném grafu je definován pouze neorientovaný sled jako posloupnost vrcholů a hran

$$v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_{k-1}, v_k$$

taková, že hrana e_i je incidentní s vrcholy v_i a v_{i+1} pro všechny $i = 1, 2, \ldots, k-1$.

Triviální sled je sled, který obsahuje jediný vrchol a žádnou hranu. Považujeme ho jak za orientovný, tak za neorientovaný.

3.1.13 Uzavřené sledy. Máme dán orientovaný (neorientovaný) sled. Říkáme, že vrchol v_1 je počátečním vrcholem sledu a v_k je koncovým vrcholem sledu. Též říkáme, že sled vede z vrcholu v_1 do vrcholu v_k.

Orientovaný (neorientovaný) sled se nazývá *uzavřený*, jestliže $v_1 = v_k$. V opačném případě mluvíme o *otevřeném* sledu.

Triviální sled nepovažujeme za uzavřený.

3.1.14 Tah, cesta. Orientovaný (neorientovaný) sled nazýváme orientovaným (neorientovaným) *tahem*, jestliže se v něm neopakují hrany.

Orientovaný (neorientovaný) tah je *cestou*, jestliže se v něm neopakují vrcholy s tou výjimkou, že může být uzavřený, tj. může být $v_1 = v_k$. Uzavřená orientovaná cesta se nazývá *cyklus*, uzavřená neorientovaná cesta se nazývá *kružnice*.

Každá cesta je zároveň tahem i sledem, naopak to ale neplatí. Také každý cyklus je zároveň kružnicí, ale ne každá kružnice je cyklem.

Poznamenejme, že triviální sled je též tahem i cestou *není* však ani kružnicí ani cyklem.

3.1.15 Dostupnost. Mějme dán graf $G = (V, E, \varepsilon)$. Řekneme, že vrchol v je *orientovaně (neorientovaně) dostupný* z vrcholu w, jestliže existuje orientovaná (neorientovaná) cesta z v do w.

Poznamejme, že jsme v definici dostupnosti mohli požadovat existenci sledu (a ne cesty) a dostali bychom stejný pojem. Když totiž existuje orientovaný (neorientovavý) sled z vrcholu v do vrcholu w, pak také existuje orientovaná (neorientovaná) cesta z vrcholu v do vrcholu w.

3.2 Souvislost

3.2.1 Souvislé grafy. Řekneme, že (orientovaný nebo neorientovaný) graf je *souvislý*, jestliže pro každé dva vrcholy u, v grafu existuje neorientovaná cesta z u do v.

Poznamenejme, že vždy existuje cesta z vrcholu u do sebe – totiž triviální cesta. Také platí, že neorientovaná cesta z vrcholu u do vrcholu v je také neorientovanou cestou z v do u.

3.2.2 Komponenty souvislosti. Máme dán (orientovaný nebo neorientovaný) graf G. *Komponenta souvislosti* (někdy též *komponenta slabé souvislosti*) je maximální množina vrcholů A taková, že indukovaný podgraf určený A je souvislý.

Maximální množinou zde rozumíme takovou množinu A, pro kterou platí, že přidáme-li k možině A libovolný vrchol, podgraf indukovaný touto větší množinou už souvislý nebude.

3.2.3 Poznámka. Graf je souvislý má-li jedinou komponentu souvislosti.

3.3 Stromy

3.3.1 Strom. Orientovaný nebo neorientovaný graf se nazývá *strom*, je-li souvislý a neobsahuje-li kružnici.

3.3.2 Tvrzení. V každém stromu s alespoň dvěma vrcholy existuje vrchol stupně 1.

3.3.3 Věta. Každý strom o n vrcholech má $n - 1$ hran.

3.3.4 Poznámka. Mějme souvislý graf G. Přidáme-li k němu hranu (aniž bychom zvětšili množinu vrcholů), zůstane graf souvislý.

Mějme graf G bez kružnic. Odebereme-li v grafu G hranu, vzniklý graf opět nebude obsahovat kružnici.

Strom je graf, který má nejmenší počet hran aby mohl být souvislý a současně největší počet hran aby v něm neexistovala kružnice.

3.3.5 Tvrzeni. Je dán graf G, pak následující je ekvivalentní.

1. G je strom.

2. Graf G nemá kružnice a přidáme-li ke grafu libovolnou hranu uzavřeme přesně jednu kružnici.

3. Graf G je souvislý a odebráním libovolné hrany přestane být souvilý.

Poznamenejme, že přidáním hrany zde rozumíme přidání hrany mezi již existující vrcholy (další vrcholy nepřidáváme).

3.3.6 Podgrafy. Je dán graf $G = (V, E, \varepsilon)$. *Podgraf* grafu G je trojice $G' = (V', E', \varepsilon')$, kde $V' \subseteq V$, $E' \subseteq E$ a ε' je restrikce ε na množině E', tak, že $G' = (V', E', \varepsilon')$ je samo grafem.

Jinými slovy, podgraf dostaneme tak, že z grafu G vynecháme některé (nebo žádné) vrcholy a některé (nebo žádné) hrany a to tak, že necháme-li v podgrafu hranu e, pak tam necháme i oba krajní vrcholy této hrany.

Podgraf $G' = (V', E', \varepsilon')$ se nazývá *faktor* grafu G, jestliže $V' = V$.

Podgraf $G' = (V', E', \varepsilon')$ je *indukovaný množinou* A, $A \subseteq V$, jestliže každá hrana, která má oba krajní vrcholy v množině A leží v E'. Podgraf indukovaný množinou A se též nazývá *úplný podgraf na množině* A.

3.4 Minimální kostra

3.4.1 Kostra grafu. Je dán souvislý graf G. Faktor grafu G, který je stromem, se nazývá *kostra* grafu G.

Připomeňme, že faktor grafu G je podgraf grafu G, který má stejnou množinu vrcholů jako G.

3.4.2 Tvrzení. Graf G má kostru právě tehdy, když je souvislý.

3.4.3 Minimální kostra. Je dán souvislý graf G spolu s ohodnocením hran c, tj. pro každou hranu $e \in E(G)$ je dáno číslo $c(e)$ (číslo $c(e)$ nazýváme *cenou hrany* e).

Minimální kostra grafu $G = (V, E)$ je taková kostra grafu $K = (V, L)$, že $\sum_{e \in L} c(e)$ je nejmenší (mezi všemi kostrami grafu G).

3.4.4 Tvrzení. V každém souvislém ohodnoceném grafu existuje minimální kostra. Nemusí však být jediná.

3.4.5 Obecný postup pro hledání minimální kostry. Je dán souvislý graf $G = (V, E)$ a ohodnocení hran c.

1. Na začátku máme $L = \emptyset$. Označíme $\mathcal{S}$ množinu všech komponent souvislosti grafu $K = (V, L)$; tj. na začátku je $\mathcal{S} = \{\{v\}; v \in V\}$.
2. Dokud není graf $K = (V, L)$ souvislý (tj. dokud $\mathcal{S}$ se neskládá z jediné množiny), vybereme hranu e podle těchto pravidel:
   (a) e spojuje dvě různé komponenty souvislosti S, S' grafu K (tj. dvě množiny z $\mathcal{S}$)
   (b) a pro S nebo S' je nejlevnější hranou, která vede z komponenty ven.

   Hranu e přidáme do množiny L a množiny S a S' nahradíme jejich sjednocením.
3. Postup ukončíme, jestliže jsme přidali $n - 1$ hran (tj. jestliže se $\mathcal{S}$ skládá z jediné množiny).

3.4.6 Tvrzení. Obecný postup 3.4.5 skončí po konečně mnoha krocích a najde některou minimální kostru.

3.4.7 Kruskalův algoritmus. Jedná se o modifikaci postupu 3.4.5:

1. Setřídíme hrany podle ceny do neklesající posloupnosti, tj.
$$c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$$
   Položíme $L = \emptyset$, $\mathcal{S} = \{\{v\}; v \in V\}$.
2. Probíráme hrany v daném pořadí. Hranu e_i přidáme do L, jestliže má oba krajní vrcholy v různých množinách $S, S' \in \mathcal{S}$. V $\mathcal{S}$ množiny S a S' nahradíme jejich sjednocením. V opačném případě hranu přeskočíme.
3. Algoritmus končí, jestliže jsme přidali $n - 1$ hran (tj. $\mathcal{S}$ se skládá z jediné množiny).

3.4.8 Primův algoritmus. Jedná se o modifikaci postupu 3.4.5:

1. Vybereme libovolný vrchol v. Položíme $L = \emptyset$, $S = \{v\}$.
2. Vybereme nejlevnější hranu e, která spojuje některý vrchol x z množiny S s vrcholem y, který v S neleží. Vrchol y přidáme do množiny S a hranu e přidáme do L.
3. Opakujeme krok 2 dokud nejsou všechny vrcholy v množině S.

3.5 Kořenové stromy

3.5.1 Kořen. Je dán orientovaný graf $G = (V, E)$. Řekneme, že vrchol $r \in V$ je *kořen* grafu G, jestliže pro každý vrchol $v \in V$ existuje orientovaná cesta z r do v.

Jinými slovy, vrchol r je kořen grafu G právě tehdy, když každý vrchol grafu G je orientovaně dostupný z vrcholu r.

Uvědomte si, že pro vrchol r existuje orientovaná cesta z r do r, totiž triviální cesta.

3.5.2 Poznámka. Každý orientovaný graf, který má kořen, je souvislý. Naopak to neplatí; existují souvislé grafy, které nemají kořen.

Orientovaný graf může mít několik kořenů; např. v cyklu je každý vrchol kořenem.

3.5.3 Kořenový strom. Orientovaný graf, který má kořen a je stromem, se nazývá *kořenový strom.*

Protože každý graf který má kořen je souvislý, mohli jsme kořenový strom definovat jako graf, který má kořen a nemá kružnice.

3.5.4 Tvrzení. Je-li G kořenový strom, pak má pouze jeden kořen.

3.5.5 Následník, předchůdce a list. Je dán kořenový strom $G = (V, E)$. Jestliže (u, v) je hrana grafu G, pak říkáme, že vrchol u je *předchůdce* vrcholu v a vrchol v je *následník* vrcholu u. Vrchol, který nemá následníka, se nazývá *list.*

3.5.6 Výška kořenového stromu. Je dán kořenový strom $G = (V, E)$ s kořenem r. Pro každý vrchol v v G existuje právě jedna orientovaná cesta z kořene r do vrcholu v. Řekneme, že vrchol v *leží v hladině* k právě tehdy, když orientovaná cesta z r do v má přesně k hran.

Výška kořenového stromu je největší k takové, že k-tá hladina je neprázdná. Jinými slovy, výška je počet hran v nejdelší orientované cestě z kořene do některého z listů.

3.5.7 Podstrom určený vrcholem. Je dán kořenový strom G. *Podstrom určený vrcholem* v je podgraf G indukovaný množinou všech orientovaně dostupných vrcholů z vrcholu v.

Uvědomme si, že podstrom určený vrcholem v je sám kořenovým stromem a jeho kořen je v.

3.5.8 Binární kořenové stromy. Kořenový strom se nazývá *binární kořenový strom*, jestliže každý vrchol má nejvýše dva následníky.

V binárním kořenovém stromě mluvíme o *pravém* a *levém následníku* vrcholu. *Levý podstrom*, resp. *pravý podstrom* vrcholu v je podstrom určený levým, resp. pravým následníkem vrcholu v.

3.5.9 Halda. Jednou z četných aplikací kořenových stromů je datová struktura zvaná *halda.* Je např. základem algoritmu Heapsort pro třídění.

3.6 Acyklické grafy

3.6.1 Orientovaný graf se nazývá *acyklický*, jestliže neobsahuje cyklus.

Poznamenejme, že acyklické grafy nemusí být souvislé a mohou obsahovat kružnici; nesmí však obsahovat smyčku.

3.6.2 Topologické očíslování vrcholů. Je dán orientovaný graf $G = (V, E, \varepsilon)$ s n vrcholy. Očíslování vrcholů

$$v_1, v_2, \ldots, v_n$$

se nazývá *topologické očíslování*, jestliže pro každou hranu e s počátečním vrcholem v_i a koncovým vrcholem v_j platí $i < j$.

Jinými slovy, hrany musí vést vždy z vrcholu s menším indexem do vrcholu s větším indexem.

3.6.3 Topologické očíslování hran. Je dán orientovaný graf $G = (V, E, \varepsilon)$ s m hranami. Očíslování hran

$$e_1, e_2, \ldots, e_m$$

se nazývá *topologické očíslování*, jestliže pro každé dvě hrany e_i, e_j pro které koncový vrchol hrany e_i je počátečním vrcholem hrany e_j platí $i < j$.

Jinými slovy, kdykoli hrana e' navazuje na hranu e, musí být v posloupnosti hrana e vypsána dříve než hrana e'.

3.6.4 Poznámka. Definici topologického očíslování hran jsme mohli formulovat i následujícím způsobem: Pro každý vrchol v platí: vypisujeme-li do posloupnosti libovolnou hranu s počátečním vrcholem v, musely již být vypsány všechny hrany, které ve vrcholu v končí.

3.6.5 Věta. Pro orientovaný graf G jsou následující podmínky ekvivalentní:

1. G je acyklický;
2. G má topologické očíslování vrcholů;
3. G má topologické očíslování hran.

3.6.6 Tvrzení. V každém acyklickém grafu existuje vrchol, který má vstupní stupeň roven 0.

3.6.7 Postup na nalezení topologického očíslování vrcholů.

1) Pro každý vrchol v spočítáme vstupní stupeň $d^-(v)$.
2) Do množiny M dáme všechny vrcholy se vstupním stupněm 0, položíme $i := 1$.
3) Dokud $M \neq \emptyset$ provedeme
   3a) Vybereme vrchol v z množiny M a odstraníme ho z M. Položíme $v_i := v$, $i := i + 1$.

3b) Pro každou hranu e s $PV(e) = v$ provedeme

$$d^-(KV(e)) := d^-(KV(e)) - 1$$

a v případě, že $d^-(KV(e)) = 0$, přidáme vrchol $KV(e)$ do množiny M.

3.6.8 Jádro grafu. Podmnožina vrcholů K orientovaného grafu G se nazývá *jádro grafu*, jestliže splňuje následující podmínky:

1. Pro každou hranu e s počátečním vrcholem $PV(e) \in K$ platí $KV(e) \notin K$. (Tj. neexistuje hrana, která by vedla z množiny K do sebe.)
2. Pro každý vrchol v, který neleží v K, existuje hrana e s $PV(e) = v$ a $KV(e) \in K$. (Tj. z každého vrcholu, který leží mimo K, se můžeme dostat po hraně zpět do K.)

3.6.9 Tvrzení. V každém acyklickém grafu existuje jádro a je určeno jednoznačně. Jádro je možné sestrojit z topologického očíslování vrcholů.

3.7 Silná souvislost

3.7.1 Silně souvislé grafy. Řekneme, že orientovaný graf G je *silně souvislý*, jestliže pro každé dva vrcholy u, v existuje orientovaná cesta z vrcholu u do vrcholu v a orientovaná cesta z vrcholu v do vrcholu u.

3.7.2 Poznámka. V definici silně souvislého grafu jsme mohli požadovat pouze existenci orientované cesty z vrcholu u do vrcholu v. Je to proto, že existenci takové cesty vyžadujeme pro všechny dvojice vrcholů, tedy i pro dvojici v, u.

Dále si uvědomte, že vždy existuje orientovaná cesta z vrcholu do sebe – je to triviální cesta.

3.7.3 Tvrzení. Souvislý graf je silně souvislý právě tehdy, když každá hrana leží v nějakém cyklu.

3.7.4 Silně souvislé komponenty. Je dán orientovaný graf G. Množina vrcholů K se nazývá *silně souvislá komponenta*, též *komponenta silné souvislosti*, jestliže podgraf indukovaný K je silně souvislý a přidáním libovolného vrcholu ke K přestane indukovaný podgraf být silně souvislý.

3.7.5 Poznámka. Předchozí definice silně souvislé komponenty se často formuluje takto: silně souvislá komponenta je maximální podmnožina vrcholů taková, že podgraf indukovaný touto množinou je silně souvislý. Termín „maximální“ zde znamená „nedá se přidat vrchol při zachování silné souvislosti“.

3.7.6 Poznámka. Každý vrchol orientovaného grafu leží přesně v jedné silně souvislé komponentě. O hranách už takové trzení neplatí – mohou existovat hrany, které neleží v žádné silně souvislé komponentě.

3.7.7 Kondenzace grafu. Je dán orientovaný graf $G = (V, E)$. *Kondenzace grafu* G je graf $\bar{G} = (\bar{V}, \bar{E})$, kde $\bar{V}$ je množina všech silně souvislých komponent grafu G a hrana vede z komponenty K_1 do komponenty K_2 právě tehdy, když $K_1 \neq K_2$ a existují vrcholy $u \in K_1$, $v \in K_2$ takové, že (u, v) je hrana grafu G.

3.7.8 Poznámka. Kondenzace grafu je vždy acyklický graf.

3.7.9 Hledání silně souvislých komponent. Silně souvislé komponenty orientovaného grafu je možné hledat Tarjanovým algoritmem, který je modifikací algoritmu porhledávání do hloubky. Modifikace spočívá v tom, že kromě pořadových čísel přiřazujeme vrcholům též číslo, které uvádí jak hluboko do prohledávacího zásobníku vede z daného vrcholu orientovaná cesta z dosud probraných hran. Na začátku příští přednášky si algoritmus uvedeme.

3.7.10 Tarjanův algoritmus pro nalezení silně souvislých komponent.

Vstup: orientovaný graf G.

Výstup: silně souvislé komponenty grafu G.

Pomocné proměnné: Každý vrchol x bude mít přiřazen dvě čísla – pořadové číslo $P(x)$ a zpětné číslo $Z(x)$. Číslo $P(x)$ udává pořadí, ve kterém byl vrchol x poprvé navštíven při prohledávání do hloubky; číslo $Z(x)$ udává nejmenší pořadové číslo vrcholu, který je z x orientovaně dostupný a to orientovanou cestou, která je tvořena několika hranami prohledávání a nejvýše jednou hranou zpět.

Dále jsou dány dva zásobníky ST_1 a ST_2 (ST_1 je zásobník prohledávání do hloubky, ST_2 je komponentový zásobník).

Každý vrchol x bude:

- Dosud nenavštíven (ještě jsme ho v prohledávání do hloubky nenavštívili).
- Již navštíven: má již určené pořadové číslo, ale ještě nebyl zařazen do komponenty – to poznáme podle toho, že je v zásobníku ST_2.
- Byl již zařazen do komponenty silné souvislosti – to poznáme podle toho, že již byl vyřazen ze zásobníku ST_2.

```
 0: i := 1; k := 1; ST_1 := ∅; ST_2 := ∅;
 1: if existuje nenavštívený vrchol x do
 2:     P(x) := i, Z(x) := P(x), vlož x na vrchol ST_1 a ST_2, i := i + 1
 3:     if existuje nepoužitá hranu e s PV(e) = x, w := KV(e)
 4:        if w ještě nebyl navštíven, then x := w a goto 2
 5:        if w je v ST_2, then
 6:           if Z(x) > P(w) then Z(x) := P(w) a goto 3
 7:        if w je již zařazen do některé komponenty then goto 3
 8:     if neexistuje nepoužitá hrana z x do
 9:        if Z(x) < P(x) do
10:           if ST_1 = ∅ , then goto 1
11:           else y := v, kde v je vrchol ST_1 a do
12:              if Z(y) > Z(x) then Z(y) := Z(x)
13:           odstraníme y ze ST_1, x := y a goto 3
14:        if Z(x) = P(x) do
15:           K_j := {u | u ∈ ST_2, P(u) ≥ P(x)}
16:           vrcholy z K_j odstraníme ze ST_2; j := j + 1
17:           if ST_1 = ∅ , then goto 1
18:           else x := v, kde v je vrchol ST_1 ,
19:              odstraníme v ze ST_1 a goto 3
```

3.7.11 Poznámky.

1. Dokázat správnost Tarjanova algoritmu není jednoduché; nejobtížnější je dokázat, že komponenty silné souvislosti se v zásobníku ST_2 nepromíchají. Důkaz jde nad rámec přednášky.
2. Tarjanův algoritmus pracuje v lineárním čase, tj. čase, který je úměrný počtu hran a počtu vrcholů.

3.8 Eulerovy grafy

3.8.1 Připomeňme, že tah je sled, ve kterém se neopakují hrany. Jinými slovy, tah obsahuje hrany grafu vždy nejvýše jedenkrát.

3.8.2 Eulerovské tahy. Tah v grafu se nazývá *eulerovský*, jestliže prochází každou hranou; jinými slovy, obsahuje-li každou hranu přesně jedenkrát. Eulerovské tahy se dělí na uzavřené a otevřené, orientované a neorientované.

3.8.3 Eulerův graf. Graf G se nazývá *eulerovský graf*, jestliže v něm existuje uzavřený eulerovský tah. V případě, že graf G je orientovaný, požadujeme existenci orientovaného uzavřeného eulerovského tahu.

3.8.4 Aplikace. Eulerovské tahy mají řadu aplikací.

- **Kreslení s co nejmenším počtem tahů.** Jedná se o tuto úlohu: Je dán souvislý graf. Úkolem je najít co nejmenší počet hranově disjunktních tahů tak, aby všechny hrany grafu byly obsaženy v některém tahu. Jinými slovy, aby tahy pokrývaly množinu všech hran grafu. Je zřejmé, že existuje-li v grafu eulerovský tah, pak je tento tah hledaným řešením.

  Řešení tohoto problému se dá využít např. při kreslení pomocí počítače (chceme co nejméně „přejezdů“).

- **Úloha čínského pošťáka.** Pošťák musí při své obchůzce projít všechny ulice. Jak to má udělat, aby ušel co nejméně kilometrů?

- **De Bruijnova posloupnost.** Je dáno přirozené číslo $k > 1$. Úkolem je najít co nejdelší cyklickou posloupnost 0 a 1 tak, aby žádné dvě po sobě následující k-tice nebyly stejné. Úloha se dá řešit nalezením uzavřeného orientovaného eulerovského tahu ve speciálním orientovaném grafu.

3.8.5 Tvrzení. V souvislém orientovaném grafu existuje uzavřený orientovaný eulerovský tah právě tehdy, když pro každý vrchol v grafu platí

$$d^-(v) = d^+(v).$$

(Tj. v každém vrcholu končí stejný počet hran jako v něm začíná.)

V souvislém grafu existuje uzavřený neorientovaný eulerovský tah přávě tehdy, když každý vrchol má sudý stupeň.

3.8.6 Postup na hledání uzavřeného orientovaného eulerovského tahu. Vybereme libovolný vrchol v grafu. Protože graf je souvislý, v každém vrcholu začíná i končí alespoň jedna hrana. Z vrcholu v vytváříme náhodně orientovaný tah; tj. procházíme hrany tak, abychom žádnou hranou neprošli dvakrát. Takto pokračujeme, dokud je to možné, tj. dokud se nevrátíme do výchozího vrcholu v a ve vrcholu v již nezačíná žadná dosud nepoužitá hrana. Tím jsme dostali uzavřený tah. Jestliže tento tah obsahuje všechny hrany, je to hledaný uzavřený eulerovský tah.

Neobsahuje-li takto zkonstruovaný tah všechny hrany, pak na tahu existuje vrchol w takový, že v něm začíná nepoužitá hrana. (To vyplývá ze souvislosti grafu.) Získaný tah ve vrcholu w rozpojíme a náhodně konstruujeme uzavřený

tah (z dosud nepoužitých hran) začínající a končící ve vrcholu w. Tento postup opakujeme, dokud nedostaneme tah obsahující všechny hrany.

3.8.7 Tvrzení. V souvislém orientovaném grafu existuje otevřený orientovaný eulerovský tah právě tehdy, když existují vrcholy u_1, u_2 takové, že

$$d^-(u_1) = d^+(u_1) + 1, \;\; d^-(u_2) = d^+(u_2) - 1,$$

a pro každý jiný vrchol v grafu platí $d^-(v) = d^+(v)$.

V souvislém grafu existuje otevřený neorientovaný eulerovský tah právě tehdy, když v grafu existují přesně dva vrcholy lichého stupně.

3.8.8 Tvrzení. Je dán souvislý neorientovaný graf G s $2k$ vrcholy lichého stupně. Pak existuje k hranově disjunktních otevřených tahů takových, že každá hrana grafu G leží v právě jednom z těchto tahů.

Ke grafu G přidáme k hran a to tak, že každá nově přidaná hrana spojuje vždy dva vrcholy lichého stupně. Tím dostaneme eulerovský graf G' (ano, každý vrchol má již sudý stupeň). V grafu G' najdeme eulerovský uzavřený tah. Jestliže z něj odstraníme všechny přidané vrcholy, rozpadne se na k hranově disjuktních tahů. Tyto tahy splňují podmínky tvrzení.

3.9 Hamiltonovské grafy

Připomeňme, že cesta je tah, ve kterém se neopakují vrcholy (s výjimkou uzavřené cesty, kdy se první vrchol rovná poslednímu).

3.9.1 Hamiltonovské cesty, kružnice, cykly. Je dán graf G. Otevřená cesta se nazývá *hamiltonovská cesta*, obsahuje-li všechny vrcholy (a tudíž všechny vrcholy přesně jedenkrát). Obdobně *hamiltonovská kružnice* je kružnice, která obsahuje každý vrchol grafu; *hamiltonovský cyklus* je cyklus, který obsahuje každý vrchol grafu.

3.9.2 Úlohy dělíme na existenční a optimalizační. V existenční úloze jde o to, zjistit zda v daném grafu existuje hamiltonovská cesta, kružnice nebo cyklus. V optimalizačních úlohách máme hrany grafu navíc ohodnoceny délkami a požaduje se nalezení hamiltonovské cesty, kružnice nebo cyklu s co nejmenším součtem délek jednotlivých hran tvořících cestu, kružnici nebo cyklus.

Na rozdíl od hledání eulerovských tahů, je hledání hamiltonovských cest, hamiltonovských kružnic a hamiltonovských cyklů velmi obtížná úloha. Přesněji, zjištění, zda v daném grafu existuje hamiltonovská cesta, kružníce nebo cyklus je tzv. NP-úplná úloha. Přesto, nebo právě proto, jsou úlohy tohoto typu v praxi rozšířené.

3.9.3 Aplikace. Uvedeme některé z aplikací hamiltonovských cest.

- **Problém obchodního cestujícího.** Jde o probém nalezení nejkratší hamiltonovské kružnice v úplném neorientovaném ohodnoceném grafu.
- **Dopravní úlohy.** Jedná se o optimalizaci pohybu nějakého dopravního prostředku; např. při rozvozu zboží, vybírání schránek apod. Často se jedná o nalezení otevřené hamiltonovské cesty. Např. při rozvozu pracovníků na roztroušená pracoviště můžeme požadovat, aby se po skončení směny dopravní prostředek nevracel prázdný, ale aby svezl pracovníky (v opačném pořadí). Hledáme proto nejkratší hamiltonovskou cestu z daného vrcholu, koncový vrchol cesty obvykle není určen.
- **Plánování procesů.** Máme nějaké výrobní zařízení na kterém se prováději procesy $p_1, p_2, \ldots, p_n$. Přitom pro některé dvojice procesů p_i, p_j platí, že má-li po skončení procesu p_i následovat proces p_j je třeba zařízení vyčistit, přestavět atd., tedy musíme zaplatit jistou cenu, aby po skončení procesu p_i mohl následovat proces p_j. Úkolem je najít takové pořadí procesů $p_1, p_2, \ldots, p_n$ aby cena byla nulová. V případě, že takové pořadí neexistuje, můžeme žádat pořadí procesů tak, aby cena byla nejmenší. Tento druhý případ vede na problém obchodního cestujícího.

3.9.4 Existují jednoduché nutné podmínky proto, aby v daném grafu existovala hamiltonovská cesta, hamiltonovská kružnice či hamiltonovský cyklus. Uvedeme několik takových tvrzení.

- Existuje-li v grafu hamiltonovská cesta, musí být graf souvislý.
- Existuje-li v grafu hamiltonovská kružnice, musí mít každý vrchol stupeň alespoň 2.

- Existuje-li v grafu G hamiltonovský cyklus, musí být graf silně souvislý.

Netriviální nutná a postačující podmínka pro zjištění, zda daný graf obsahuje hamiltonovskou cestu, kružnici nebo cyklus, není známa.

3.9.5 Metoda větví a mezí. Potřebujeme vyřešit optimalizační úlohu $\mathcal{U}$, která je dána podmínkami P a účelovou funkcí c. Přípustné řešení je každé řešení, které splňuje podmínky P, optimální je pak to přípustné řešení, které má nejlepší hodnotu účelové funkce c. Množinu přípustných řešení úlohy $\mathcal{U}$ budeme značit $PR(\mathcal{U})$.

Podmínky P rozdělíme na „jednoduché“ podmínky — A a „složíté“ podmínky — B. Jako $\tilde{\mathcal{U}}$ označíme optimalizační úlohu, která je dána pouze podmínkami A a stejnou účelovou funkcí c.

Nejprve řešíme úlohu $\tilde{\mathcal{U}}$ (tj. zapomeneme na podmínky B — říkáme, že podmínky B relaxujeme) a najdeme optimální řešení *opt* úlohy $\tilde{\mathcal{U}}$. Hodnota účelové funkce pro řešení *opt* nám slouží jako odhad, jaké „nejlepší“ řešení úloha $\mathcal{U}$ (tedy úloha daná podmínkami P) může mít. Splňuje-li nalezené řešení *opt* i podmínky B (tj. je-li *opt* přípustné řešení úlohy $\mathcal{U}$), dostali jsme optimální řešení úlohy $\mathcal{U}$.

Jestliže řešení *opt* nesplňuje podmínky B, rozdělíme úlohu $\mathcal{U}$ (*větvíme úlohu*) na podúlohy $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_k$ a to tak, že

1. $k \geq 2$ a
2. každé přípustné řešení úlohy $\mathcal{U}$ je přípustným řešením některé z podúloh $\mathcal{U}_1, \ldots, \mathcal{U}_k$. Tj.

$$PR(\mathcal{U}) = PR(\mathcal{U}_1) \cup \ldots \cup PR(\mathcal{U}_k).$$

Podmínka 2 nám zaručuje, že optimální řešení úlohy $\mathcal{U}$ bude některé z optimálních řešení úloh $\mathcal{U}_1, \ldots, \mathcal{U}_k$.

Jestliže úloha $\mathcal{U}_i$ již nemá přípustné řešení, nebo odhad pro optimální řešení je horší nebo stejný jako již známé přípustné řešení, úlohou se dále nezabýváme (*úlohu umrtvíme*). V opačném případě řešíme $\mathcal{U}_i$ stejným způsobem.

Výpočet ukončíme tehdy, když pro všechny podúlohy jsme buď našli optimální řešení, nebo jsme je umrtvili.

3.9.6 Poznámka. Pro případ hledání nejlevnější hamiltonovské cesty C máme:

- A – hamiltonovská cesta je kostra,
- B – každý vrchol v hamiltonovské cestě má stupeň nejvýše 2,
- účelová funkce je součet cen jednotlivých hran v C,
- cena minimální kosty slouží jako odhad optimálního řešení dané podúlohy.

3.9.7 Pro zmenšení počtu větvení můžeme použít ještě jiný způsob větvení úlohy než je uveden výše.

Označme v vrchol, který má v minimální kostře stupeň $d(v) = k > 3$ a označme e_1, $e_, \ldots, e_k$ hrany, s krajním vrcholem v. Úlohu $\mathcal{U}$ rozdělíme ma tři podúlohy $\mathcal{U}_1, \mathcal{U}_2$ a $\mathcal{U}_3$ takto:

$\mathcal{U}_1$ — zakážeme hranu e_1;

$\mathcal{U}_2$ — vynutíme hranu e_1 a zakážeme hranu e_2;

$\mathcal{U}_3$ — vynutíme hrany e_1 a e_2 a zakážeme všechny hrany e_3, $e_4, \ldots, e_k$.

Nyní platí

$$PR(\mathcal{U}) = PR(\mathcal{U}_1) \cup PR(\mathcal{U}_2 \cup PR(\mathcal{U}_3$$

a navíc množiny připustných řešení úloh $\mathcal{U}_1, \mathcal{U}_2$ a $\mathcal{U}_3$ jsou po dvou disjunktní.

3.10 Nezávislost, kliky a barevnost grafu

3.10.1 Nezávislé množiny. Je dán neorientovaný (orientovaný) graf G. Množina vrcholů A se nazývá *nezávislá množina vrcholů*, jestliže žádná hrana grafu G nemá oba krajní vrcholy v množině A. Jinými slovy, podgraf indukovaný množinou A je diskrétní.

3.10.2 Maximální nezávislá množina. Je dán graf G. Nezávislá množina N se nazývá *maximální nezávislá množina*, jestliže jakákoli její nadmnožina už není nezávislá.

Jinými slovy, N je maximální nezávislá množina, jestliže pro každý vrchol v, který neleží v N, existuje vrchol $w \in N$ takový, že v G existuje hrana mezi v a w.

3.10.3 Nezávislost grafu. Je dán neorientovaný nebo orientovaný graf G. Počet vrcholů v nejpočetnější nezávislé množině grafu G se nazývá *nezávislost* grafu G a značíme jej $\alpha(G)$.

Nejpočetnější nezávislá množina je jistě také maximální, ale ne každá maximální nezávislá množina je současně nejpočetnější.

3.10.4 Poznámka. Jádro grafu orientovaného grafu G definované v 3.6.8 je nezávislá množina grafu G; to vyplývá z první podmínky, kterou jádro musí splňovat. Ovšem ne každá nezávislá množina orientovaného grafu G je současně jádrem grafu G; jádro musí ještě splňovat i druhou podmínku z 3.6.8.

3.10.5 Úplný neorientovaný graf. Neorientovaný graf G nazýváme *úplným grafem*, jestliže je prostý, nemá smyčky a každé dva různé vrcholy jsou spojené hranou.

Úplný neorientovaný graf G s n vrcholy má $\frac{n(n-1)}{2}$ hran.

3.10.6 Klika v grafu. Je dán neorientovaný graf G. Množina vrcholů K grafu G se nazývá *klika* v grafu G, jestliže každé dva různé vrcholy z množiny K jsou spojeny hranou a je maximální s touto vlastností.

Jinými slovy, podgraf určený množinou vrcholů K je úplný a kdykoli v je vrchol, který neleží v množině K, tak v K existuje vrchol w, který s v spojen hranou není.

3.10.7 Doplňkový graf. Je dán neorientovaný graf $G = (V, E)$ bez smyček. Pak *doplňkový graf* grafu G je graf $G^{dop} = (V, E^{dop})$, kde

$$\{u, v\} \in E^{dop} \quad \text{právě tehdy, když} \quad u \neq v \ \text{ a } \ \{u, v\} \notin E.$$

3.10.8 Tvrzení. Množina N vrcholů grafu G je maximální nezávislá množina grafu G právě tehdy, když je to klika v doplňkovém grafu G^{dop}.

Množina K vrcholů grafu G je klika v grafu G právě tehdy, když je to maximální nezávislá množina doplňkového grafu G^{dop}.

3.10.9 Obarvení grafu. Je dán neorientovaný graf G bez smyček. *Obarvení* vrcholů grafu je přiřazení barev (prvků množiny B) vrcholům grafu G a to takovým způsobem, že žádné dva vrcholy spojené hranou nemají stejnou barvu.

3.10.10 k-barevný graf. Graf G se nazývá *k-barevný*, jestliže se dá obarvit k barvami (tj. jestliže lze graf obarvit tak, aby množina barev B měla k prvků).

3.10.11 Barevnost grafu. Je dán neorientovaný graf G bez smyček. *Barevnost* grafu G (též *chromatické číslo* grafu G) je nejmenší k takové, že G je k-barevný. Barevnost grafu G značíme $\chi(G)$.

3.10.12 Pozorování. Množina vrcholů obarvená stejnou barvou tvoří nezávislou množinu grafu.

Graf je jednobarevný právě tehdy, když nemá žádnou hranu.

3.10.13 Tvrzení. Graf G je dvoubarevný právě tehdy, když neobsahuje kružnici liché délky.

3.10.14 Dvoubarevné grafy. Zjistit, zda je daný graf dvoubarevný, se dá jednoduchou modifikací prohledávání do šířky:

Provedeme prohledání grafu do šířky. Vrcholům, které ležely v sudých hladinách, přiřadíme barvu 1; vrcholům, které ležely v lichých hladinách, přiřadíme barvu 2.

Jestliže graf neobsahoval kružnici liché délky, jedná se o obarvení grafu a graf je tedy dvoubarevný. Vede-li hrana mezi dvěma vrcholy v hladinách stejné parity, obsahuje graf kružnici liché délky a není proto dvoubarevný.

3.10.15 Bipartitní grafy. Graf G se nazývá *bipartitní*, jestliže jeho množinu vrcholů můžeme rozdělit na dvě množiny X a Y (tj. $V(G) = X \cup Y$ a $X \cap Y = \emptyset$) tak, že každá hrana má jeden krajní vrchol v množině X a druhý krajní vrchol v množině Y.

3.10.16 Tvrzení. Graf G je bipartitní právě tehdy, když je dvoubarevný.

3.10.17 Poznámka. Zjistit, zda daný graf je tříbarevný, je těžký problém (obecně NP-úplný problém).

3.10.18 Tvrzení. Pro každý prostý graf G, který má m hran platí

$$\chi(G) \leq \frac{1}{2} + \sqrt{2m + \frac{1}{4}}.$$

3.10.19 Tvrzení. Označme Δ největší stupeň vrcholu grafu G. Pak

$$\chi(G) \leq \Delta + 1.$$

3.10.20 Sekvenční barvení. Následující postup obarví graf $\Delta + 1$ barvami. Označme množinu barev $B = \{1, \ldots, \Delta + 1\}$.

1. Seřadíme vrcholy do posloupnosti (libovolně)

$$v_1, v_2, \ldots, v_n$$

2. Probíráme vrcholy v tomto pořadí a vrcholu v_i přiřadíme vždy tu nejmenší barvu, kterou nemá žádný jeho soused vrcholu.

3.10.21 Poznámka. Algoritmus sekvenčního barvení dává horní odhad pro barevnost grafu. Jedná se ovšem o odhad, který může být velmi vzdálen od barevnosti grafu. Přesněji, existují dvoubarevné grafy, které při nevhodném uspořádání vrcholů v kroku 1, algoritmus obarví $\frac{n}{2}$ barvami (kde n je počet vrcholů grafu).

3.10.22 Poznámka. Tvrzení 3.10.19 je možné modifikovat takto:

Každý prostý graf G splňuje

$$\chi(G) \leq 1 + \max\{\delta(H) \mid H \subseteq G\},$$

kde $\delta(H)$ je rovno nejmenšímu stupni vrcholu v grafu H.

3.10.23 Tvrzení. Pro každý neorientovaný graf G bez smyček platí:

$$\alpha(G) + \chi(G) \leq n + 1$$

kde n je počet vrcholů grafu G.

Připomeňme, že $\alpha(G)$ je nezávislost grafu G, tj. počet vrcholů v nejpočetnější nezávislé množině grafu G.